UNIVERSITY OF OSLO
Department of Informatics

Probabilistic Real-Time Rewrite Theories and Their Expressive Power

Research Report 430

Lucian Bentea

Peter Csaba Ölveczky

ISBN 82-7368-395-8
ISSN 0806-3036

June 17, 2014

Probabilistic Real-Time Rewrite Theories and Their Expressive Power

Lucian Bentea and Peter Csaba Ölveczky

Department of Informatics, University of Oslo

Abstract. Unbounded data structures, advanced functions and data types, and/or different forms of communication are often needed to model large and complex probabilistic real-time systems such as wireless sensor network algorithms. Furthermore, it is natural to model distributed probabilistic real-time systems in an object-oriented style, including using subclass inheritance and dynamic object and message creation and deletion. To support the above features, we introduce probabilistic real-time rewrite theories (PRTRTs), that extend both real-time rewrite theories and probabilistic rewrite theories, as a rewriting-logic-based formalism for probabilistic real-time systems. We then show that PRTRTs can be seen as a unifying model in which a range of other models for probabilistic real-time systems—including probabilistic timed automata, stochastic automata, deterministic and stochastic Petri nets, as well as two probabilistic timed transition system models with underspecified probability distributions—can naturally be represented. We also provide semantics-preserving mappings from these models into PRTRTs, and prove their correctness. Finally, we show how the OGDC state-of-the-art algorithm for wireless sensor network algorithm can be specified in our formalism.

1 Introduction

Many large distributed systems, such as network communication protocols and wireless sensor network algorithms, are real-time systems that exhibit probabilistic behaviors. Because of their size and complexity, such systems may not be easily modeled (if at all) using model checking tools whose specification formalisms sacrifice expressiveness for decidability of key properties.

In this report we therefore introduce a new formalism, *probabilistic real-time rewrite theories* (PRTRTs), that extends rewriting logic [23] to support the formal specification of probabilistic real-time systems. Rewriting logic is a logic for concurrent systems that emphasizes expressiveness and ease of specification over algorithmic decidability of key properties. In rewriting logic, the state space and data types of a system are defined by an algebraic equational specification, and the system's transitions are defined by labeled conditional rewrite rules $l : t \longrightarrow t'$ **if** *cond*, where t and t' are terms that may contain universally quantified variables. Rewriting logic supports the specification of any computable data type, and distributed systems can be naturally modeled in an object-oriented style, with class inheritance and dynamic creation and deletion of objects and messages. Simulation, reachability analysis, and LTL model checking for rewriting logic is provided by the high-performance Maude tool [10]. (Since properties are in general undecidable, these analyses may not always terminate.)

The Real-Time Maude tool [27] and its underlying real-time rewrite theory formalism [26] extend rewriting logic and Maude to the formal modeling and analysis of real-time systems. Its expressiveness has made it possible to apply the tool to several large applications (see [28] for an overview) that are beyond the scope of most model checkers for real-time systems. However, some of those applications, including the LMST and OGDC wireless sensor network algorithms [16, 30] and the AER/NCA and NORM multicast protocols [29, 21], include probabilistic features—e.g., nodes may exhibit random behavior by design to break symmetries in a network, or the environment may interact with the system in a probabilistic manner—that can only be treated in an *ad hoc* way in Real-Time Maude.

Rewriting logic has also been extended to *probabilistic rewrite theories* to specify probabilistic behaviors [18]. Probabilistic rewrite theories combine nondeterministic and probabilistic behaviors, and the main idea is that the variables in the righthand side t' of a rewrite rule that do not occur in the lefthand side t are instantiated probabilistically. The VeStA tool [31] can be used for both statistical model checking and estimating numerical values in such theories, and has been used to analyze a DoS resistant TCP/IP protocol [1] and a model of the above mentioned LMST algorithm [16] in which its real-time behavior is treated in an *ad hoc* way.

As expected, PRTRTs can be seen as an extension of both real-time rewrite theories and probabilistic rewrite theories. However, PRTRTs are a *proper* extension of probabilistic rewrite theories even when time is ignored. In our case, the new variables in the righthand side of a rule are divided into *nondeterministic*

and *probabilistic* variables. Each rewrite rule is equipped with a *family* of probability distributions used to instantiate the probabilistic variables; namely, there is one probability distribution for each substitution of the variables in the lefthand side of the rule *and* each choice of values for the nondeterministic variables. Regarding time, although the duration of/between events may be given probabilistically, time itself does not advance in a probabilistic way in our formalism.

After giving some background into rewriting logic and its probabilistic and real-time extensions (Section 2), we define our formalism and its semantics (Section 3) and the probability of reaching a certain state in a certain time (Section 5). We then demonstrate the expressiveness of PRTRTs—as well as their suitability as a *unifying semantic framework* for probabilistic real-time systems in which different models for such systems can naturally be represented and understood—by showing how a range of well known formal models for probabilistic real-time systems can be seen as PRTRTs (Section 6). This section includes formal proofs of correctness for the representations. Appendix B also gives a more detailed specification of the PRTRT representation of deterministic and stochastic Petri nets (DSPNs), using Maude syntax. To illustrate our formalism, we describe how a simple probabilistic round trip time protocol can be modeled as a PRTRT (Section 4). This simple system cannot be modeled as an automaton, since the number of messages in the state can grow beyond any bound. Appendix A gives the full PRTRT specification of this protocol, using Maude syntax.

Finally, in Section 7 we explain how the state-of-the-art OGDC algorithm [36] for wireless sensor networks can be defined as a PRTRT. A main feature of the algorithm is that a sensor node becomes *active* depending on how close it is to an "ideal" position w.r.t. the already active nodes, and that it turns itself off to save energy when its sensing area is covered by the sensing areas of other active nodes. The OGDC algorithm therefore requires computing with data types for sensing areas and sophisticated functions including distances, angles, computing overlaps of areas, etc., which clearly seem to be beyond the capability of formalisms that do not support the definition of new data types and advanced functions.

2 Background and Notation

In rewriting logic [23], the static parts of a system (functions, data types, etc.) are defined as an algebraic equational specification, and the transitions of a system are specified by labeled rewrite rules of the form $l: \ t \longrightarrow t' \ \ \textbf{if} \ \ cond$, where t and t' are *terms* constructed by typed variables and function symbols in a type-consistent way, l is a rule label, and *cond* is a (possibly empty) conjunction of equalities, sort memberships, and rewrites. Such a rule specifies a local transition from an instance of the term t to the corresponding instance of the term t', provided that the condition *cond* is satisfied by the substitution instance.

Formally, let K be a set whose elements are called *kinds* and denote by K^* the free monoid on K. A *K-kinded signature* is a pair (K, Σ), where $\Sigma = \{\Sigma_{w,k} \mid w \in K^*, \ k \in K\}$ is a $K^* \times K$-indexed family of sets of *function symbols*, with an element $f \in \Sigma_{w,k}$ denoted $f : w \to k$. A *MEL signature* is a triple (K, Σ, S), also denoted Σ, with (K, Σ) a K-kinded signature and $S = \{S_k \mid k \in K\}$ a K-indexed family of disjoint sets of *sorts*.

Definition 1. *A* membership equational logic (MEL) theory *[24] is a pair (Σ, E) consisting of a MEL signature Σ, together with a set of axioms E, which are either* conditional Σ-memberships *of the form*

$$(\forall \vec{x}) \quad t \ : \ s \quad \textbf{if} \quad u_1 = v_1 \ \wedge \ \ldots \ \wedge \ u_n = v_n \quad \wedge \quad w_1 : s_1 \ \wedge \ \ldots \ \wedge \ w_m : s_m,$$

or conditional Σ-equalities *of the form*

$$(\forall \vec{x}) \quad t = t' \quad \textbf{if} \quad u_1 = v_1 \ \wedge \ \ldots \ \wedge \ u_n = v_n \quad \wedge \quad w_1 : s_1 \ \wedge \ \ldots \ \wedge \ w_m : s_m,$$

where t and t' are Σ-terms of the same kind $k \in K$, s is a sort of kind k, u_i and v_i are terms of the same kind, and $w_j : s_j$ asserts that the term w_j must be of sort s_j. Furthermore, $\vec{x}$ is the set of variables occurring in t, t', u_i, v_i and w_j, for all $i \in \{1, \ldots, n\}$ and all $j \in \{1, \ldots, m\}$.

We write $vars(t)$ for the set of variables occurring in a term t; if $vars(t) = \emptyset$, then t is called a *ground term*. If A is a collection of *structural axioms* specifying properties of function symbols, like commutativity, associativity, etc., then a MEL theory $(\Sigma, E \cup A)$ generates an *initial algebra* $T_{\Sigma/(E \cup A)}$; if E is terminating, confluent and sort-decreasing modulo A [8], $T_{\Sigma/(E \cup A)}$ is isomorphic to the algebra $Can_{\Sigma,E/A}$ of fully simplified ground terms, or "normal forms," with respect to the set of axioms E, modulo the structural axioms A. We denote by $[t]_A$ the A-equivalence class of a fully simplified term t.

Definition 2. *An* E/A-canonical ground substitution *for a set of variables* $\vec{x}$ *is a function* $[\theta]_A : \vec{x} \to Can_{\Sigma,E/A}$ *that assigns a fully simplified ground term to each variable in* $\vec{x}$*; we denote by* $CanGSubst_{E/A}(\vec{x})$ *the set of all such functions. We also write* $[\theta]_A$ *for the homomorphic extension* $[\theta]_A : T_\Sigma(\vec{x}) \to Can_{\Sigma,E/A}$ *of an* E/A*-canonical ground substitution* $[\theta]_A$ *to* Σ*-terms.*

A *generalized MEL signature* is a pair (Σ, φ), where Σ is a MEL signature and φ is a function that maps each function symbol $f : k_1 \dots k_n \to k$ in Σ to the set $\varphi(f) \subseteq \{1, \dots, n\}$ of its *frozen argument positions.*

Definition 3. *A* generalized rewrite theory *[9] is a tuple* $\mathcal{R} = (\Sigma, \varphi, E, L, R)$*, where* (Σ, φ) *is a generalized MEL signature,* (Σ, E) *is a MEL theory, and* R *is a set of* labeled conditional rewrite rules *of the form*

$$(\forall \vec{x}) \;\; l : \; t \longrightarrow t' \;\; \textbf{if} \;\; cond, \tag{1}$$

where $l \in L$ *is a label,* t *and* t' *are terms of the same kind,* $cond$ *is a conjunction of equalities, memberships and rewrites, and* $\vec{x} = vars(t) \cup vars(t') \cup vars(cond)$*.*

Intuitively, if i is a frozen position of a function symbol f, i.e., if $i \in \varphi(f)$, then $f(\dots, t_i, \dots)$ does *not* rewrite to $f(\dots, t'_i, \dots)$ when t_i rewrites to t'_i. The (standard) rewrite theories [23] are generalized rewrite theories with $\varphi(f) = \emptyset$ for each function symbol f. A *context* [18] is a Σ-term $\mathbb{C}$ with a single occurrence of a single variable, denoted $\odot$ and called the *hole*. Two contexts $\mathbb{C}$ and $\mathbb{C}'$ are called A*-equivalent* if $A \vdash (\forall \odot)\ \mathbb{C}(\odot) = \mathbb{C}'(\odot)$, and we write $[\mathbb{C}]_A$ for the A-equivalence class of $\mathbb{C}$. We say that a context $f(t_1, \dots, t_n)$ has a hole in a frozen position if the hole occurs in some argument t_i, and either $i \in \varphi(f)$ or t_i has a hole in a frozen position.

Let I be a finite or countably infinite set. Given a set $\Omega \neq \emptyset$, a σ*-algebra* over Ω is a collection of sets $\mathcal{F} \subseteq \mathcal{P}(\Omega)$ such that $\Omega \setminus F \in \mathcal{F}$ for all $F \in \mathcal{F}$, and $\bigcup_{i \in I} F_i \in \mathcal{F}$ for all collections $\{F_i\}_{i \in I} \subseteq \mathcal{F}$. Let $\mathcal{F}$ be a σ-algebra over Ω. An $\mathcal{F}$*-cover* is a function $\alpha : \Omega \to \mathcal{F}$ satisfying $\omega \in \alpha(\omega)$, for all $\omega \in \Omega$. A function $\mathbb{P} : \mathcal{F} \to [0, 1]$ is called a *probability measure* if $\mathbb{P}(\Omega) = 1$ and $\mathbb{P}\left(\cup_{i \in I} F_i\right) = \sum_{i \in I} \mathbb{P}(F_i)$, for all collections $\{F_i\}_{i \in I} \subseteq \mathcal{F}$ of pairwise disjoint sets. The triple $(\Omega, \mathcal{F}, \mathbb{P})$ is then called a *probability space*. Denote by $PMeas(\Omega, \mathcal{F})$ the set of all probability measures defined on the σ-algebra $\mathcal{F}$ over Ω. A function $p : \Omega \to [0, 1]$ with the property that $\sum_{\omega \in \Omega} p(\omega) = 1$ is called a *probability mass function*, or *probability distribution*. If Ω is finite or countably infinite, a probability mass uniquely defines a probability measure $\mathbb{P} : \mathcal{F} \to [0, 1]$ via $\mathbb{P}(A) = \sum_{a \in A} p(a)$, for all sets $A \in \mathcal{F}$. We denote by $PMF(\Omega)$ the set of all probability mass functions on Ω. In the case when $\Omega = \mathbb{R}$, we consider $\mathcal{F}$ to be the standard Borel σ-algebra over $\mathbb{R}$, also denoted $\mathcal{B}_\mathbb{R}$, which is the smallest σ-algebra over $\mathbb{R}$ that contains all the open intervals. Let $\varphi : \mathbb{R} \to [0, 1]$ be a monotonically increasing and right-continuous function such that $\lim_{t \to -\infty} \varphi(t) = 0$ and $\lim_{t \to \infty} \varphi(t) = 1$. Each such function defines a probability measure $\mathbb{P} \in PMeas(\mathbb{R}, \mathcal{B}_\mathbb{R})$ via $\mathbb{P}((-\infty, x)) = \varphi(x)$, i.e., $\varphi(x)$ is the probability of selecting a real number which is less than some $x \in \mathbb{R}$. The function φ is also known as a *cumulative distribution function* (CDF); we denote by $CDF(\mathbb{R})$ the set of all CDFs.

In [18] rewrite theories are extended to *probabilistic rewrite theories*. Intuitively, in such theories the righthand side t' of a rewrite rule $l : \; t \longrightarrow t' \;\; \textbf{if} \;\; cond$ may contain variables $\vec{p}$ that do not occur in t. These new variables are assigned values according to a probability measure taken from a *family* of probability measures—one for each instance of the variables in t—associated with the rule. Formally, a probabilistic rewrite theory is a pair $(\mathcal{R}, \pi)$, where $\mathcal{R}$ is a (standard) rewrite theory and π is a function which assigns to each rule $r \in R$ of the form (1), with $vars(t) = \vec{x}$ and $vars(t') \setminus vars(t) = \vec{p}$, a mapping

$$\pi_r : [\![cond(\vec{x})]\!] \to PMeas\left(CanGSubst_{E/A}(\vec{p}),\ \mathcal{F}_r\right),$$

where $[\![cond(\vec{x})]\!]$ is the set of all E/A-canonical ground substitutions for $\vec{x}$ that satisfy the condition $cond$, and $\mathcal{F}_r$ is a σ-algebra on $CanGSubst_{E/A}(\vec{p})$. That is, for each substitution θ of the variables in t which satisfies $cond$, we get a probability measure $\pi_r\left([\theta]_A\right) : \mathcal{F}_r \to [0, 1]$ that defines how the new variables $\vec{p}$ are instantiated, i.e., the probability of selecting one of the substitutions in a set $S \subseteq CanGSubst_{E/A}(\vec{p})$ is given by $\pi_r\left([\theta]_A\right)(S)$. A labeled conditional rewrite rule $r \in R$ of the form (1) with $vars(t') \setminus vars(t) \neq \emptyset$, together with its associated probability distribution function π_r, is called a *probabilistic rewrite rule* and is written:

$$l : \; t \longrightarrow t' \;\; \textbf{if} \;\; cond \;\; \textbf{with probability} \;\; \pi_r$$

In [26], rewrite theories are extended to real-time systems by *(i)* considering ordinary rewrite rules to define *instantaneous* transitions, and *(ii)* by adding *tick rewrite rules* that model time elapse in a system. Formally, a *real-time rewrite theory* [26] is a triple $(\mathcal{R}, \phi, \tau)$ where:

– $\mathcal{R}$ is a generalized rewrite theory whose signature Σ contains a sort `System`, denoting the current state of the system, as well as a special sort `GlobalSystem` with no subsorts or supersorts, and which does not appear in the arity of any function symbol in Σ;
– the signature Σ contains an operation `{_} : System -> GlobalSystem` that satisfies no non-trivial equations, and is used for enclosing the entire system state;
– ϕ interprets the theory TIME [26] in $\mathcal{R}$;
– τ is an assignment of a *duration* term τ_l of sort ϕ(Time) to rewrite rules of the form

$$l : \{t\} \longrightarrow \{t'\} \textbf{ if } cond$$

whose both sides are terms of sort `GlobalSystem`. The term τ_l specifies the amount of time that elapses with the application of the rewrite rule; if $\tau_l \neq \phi(0)$, the rule is called a *tick rewrite rule*. A rewrite rule of the form (1), together with its associated duration term τ_l is written

$$(\forall \vec{y}) \;\; l : \; \{t\} \xrightarrow{\tau_l} \{t'\} \;\; \textbf{if} \;\; cond \tag{2}$$

where $\vec{y}$ is the set of all variables occurring in t, t' and *cond*, together with the variables in τ_l.

We also use the Maude [10] syntax to specify rewrite rules, so that a conditional tick rule with duration y is written `crl [`l`]: {`t`} => {`t'`} in time` y `if` *cond*, where the label l may be omitted. In object-oriented Maude specifications [10], the state of the system is a term of sort `Configuration` denoting a multiset of objects and messages, where multiset union is denoted by juxtaposition. Each object is represented as a term

$$\texttt{<}\ o : c \mid att_1 : val_1,\ \ldots,\ att_n : val_n\ \texttt{>}$$

where o is the object's identifier of sort `Oid`, c is the object's class, and where $val_1, \ldots, val_n$ are the values of the object's attributes $att_1, \ldots, att_n$. We use the `dly` operator in [27] to model message transmission delays, where `dly(m, x)` denotes that message `m` will become available for consumption in time `x`. For example, the following rule

```
rl [l]: m(O, w)  < O : C | a1 : x, a2 : O', a3 : z >   =>
                 < O : C | a1 : x + w, a2 : O', a3 : z >  dly(m'(O'), x) .
```

defines a family of transitions in which a message `m`, with parameters `O` and `w`, is read and consumed by an object `O` of class `C`. The transitions change the attribute `a1` of `O` and send a new message `m'(O')` with delay `x`. "Irrelevant" attributes (such as `a3` and the righthand side occurrence of `a2`) need not be mentioned in a rule.

3 Probabilistic Real-Time Rewrite Theories

This section defines probabilistic real-time rewrite theories (PRTRTs) and their semantics. PRTRTs extend both probabilistic rewrite theories and real-time rewrite theories to support the formal specification of real-time systems with probabilistic features. The definitions in this section are mostly extensions of similar definitions in [18] for (untimed) probabilistic rewrite theories.

Definition 4. *A* probabilistic real-time rewrite theory *(PRTRT) is a tuple* $\mathcal{R}_{\pi,\phi,\tau} = (\mathcal{R}, \pi, \phi, \tau)$*, where* $\mathcal{R} = (\Sigma, \varphi, E \cup A, L, R)$ *is a generalized rewrite theory in which the rules in* R *have no rewrites in their conditions,* $(\mathcal{R}, \phi, \tau)$ *is a real-time rewrite theory, and* π *is a function that takes each rewrite rule* $r \in R$ *of the form (2), with* $vars(t) = \vec{x}$ *and* $vars(t') \setminus vars(t) = \vec{y} \uplus \vec{p}$ *such that* $\vec{p} \cap vars(\tau_l) = \emptyset$ *(but* $\vec{y} \cap vars(\tau_l)$ *may be nonempty), and assigns to it a mapping*[1]

$$\pi_r : [\![cond(\vec{x} \cup \vec{y} \cup vars(\tau_l))]\!] \to PMeas\left(CanGSubst_{E/A}(\vec{p}),\ \mathcal{F}_r\right)$$

such that, for each substitution $[\theta]_A \in CanGSubst_{E/A}(\vec{x} \cup \vec{y} \cup vars(\tau_l))$ *that satisfies the condition cond,* $\pi_r([\theta]_A)$ *is a probability measure on a* σ*-algebra* $\mathcal{F}_r$ *over the set of ground substitutions* $CanGSubst_{E/A}(\vec{p})$*. Probabilistic tick rewrite rules are written:*

$$l : \; \{t\} \xrightarrow{\tau_l} \{t'\} \;\; \textbf{if} \;\; cond \;\; \textbf{with probability} \;\; \pi_r.$$

[1] The mapping π_r in this definition is more general than the one in the definition of PRTRTs proposed in [7], since it allows the probability measures to also depend on the variables in the duration term τ_l.

Although the duration of/between events may be given probabilistically, time itself does not advance in a probabilistic way. The duration term τ_l therefore does not contain variables that are substituted probabilistically, hence $\vec{p} \cap vars(\tau_l) = \emptyset$ in the assumptions of the previous definition. Apart from adding timed behaviors, PRTRTs also extend probabilistic rewrite theories in two ways that are necessitated by the way tick rules are usually defined:

1. PRTRTs allow new variables that are not assigned a probability distribution in the righthand side of a rewrite rule. The main reason is that tick rules—in particular for dense time domains—allow time to advance by *any* amount less than a certain bound. Therefore, in their encodings as (untimed) rewriting logic rules via the clocked representation [26], tick rules have a new (non-probabilistic) variable in their righthand sides that defines the duration of the rewrite. Apart from representing the duration of a transition, the new variables in the righthand side of a rule may also allow to nondeterministically modify other system parameters when making the transition.

2. The tick rules involve functions on the state, such as a function defining the effect of time elapse on a state, that are *frozen* operators; we therefore have used *generalized* rewrite theories as the underlying formalism.

Example 1. Consider the following probabilistic tick rule r (written in an intuitive way):

$$\{f(x)\} \xrightarrow{y} \{g(x,y,u,z_1,z_2)\} \;\; \textbf{if } y \leq 10 \;\wedge\; u \leq 1$$
$$\textbf{with probability } \; z_1 := \begin{pmatrix} h(x,y) & f(y,u) \\ 1 - u * y/10 & u * y/10 \end{pmatrix} \;\; \textbf{and } \; z_2 := \begin{pmatrix} 0 & 1 \\ 1/2 & 1/2 \end{pmatrix}.$$

The righthand side term $\{g(x,y,u,z_1,z_2)\}$ contains variables y, u, z_1 and z_2 that do not occur in the rule's lefthand side $\{f(x)\}$. Let $\{f(t)\}$ be the state of the system when the rule is applied. The variable y is then instantiated *nondeterministically* with any value t' less than or equal to 10. Similarly, u is instantiated *nondeterministically* with any value $\alpha \in [0,1]$. The variables z_1 and z_2 are then instantiated probabilistically, where z_1 is assigned the value $[h(t,t')]_A$ with probability $1 - \alpha\, t'/10$ and the value $[f(t',\alpha)]_A$ with probability $\alpha\, t'/10$. Formally, the mapping

$$\pi_r : CanGSubst_{E/A}(\{x,y,u\}) \to CanGSubst_{E/A}(\{z_1,z_2\})$$

associated to the above rule r is given by

$$\pi_r([\theta]_A)(\{z_1 \mapsto [h(\theta(x),\theta(y))]_A,\; z_2 \mapsto i\}) = 1/2 - [\![\theta(u)]\!] \cdot [\![\theta(y)]\!] \,/\, 20$$
$$\pi_r([\theta]_A)(\{z_1 \mapsto [f(\theta(y),\theta(u))]_A,\; z_2 \mapsto i\}) = [\![\theta(u)]\!] \cdot [\![\theta(y)]\!] \,/\, 20$$

for $i \in \{[0]_A, [1]_A\}$, where $[\![t]\!]$ denotes the unique real number associated with the term t.

Let $\mathcal{R}_{\pi,\phi,\tau} = (\Sigma, \varphi, E \cup A, L, R, \pi, \phi, \tau)$ be a PRTRT. Intuitively, an R/A-match contains the complete information on how and in which context the current system state is matched against a particular rewrite rule in the specification of that system. We extend the definition of R/A-matches in [18] as follows:[2]

Definition 5. *Given a fully simplified term $[u]_A \in Can_{\Sigma,E/A}$, its* generalized R/A-matches *are triples $([\mathbb{C}]_A, r, [\theta]_A)$ where:*

1. *$\mathbb{C}$ is a context whose hole is not in a frozen position;*
2. *$r \in R$ is a rewrite rule together with its associated duration term τ_l, of the form (2);*
3. *$[\theta]_A \in CanGSubst_{E/A}(\vec{x} \cup \vec{y} \cup vars(\tau_l))$ is a substitution such that $E \cup A \vdash \theta(cond)$;*
4. *$[u]_A = [\mathbb{C}(\odot \leftarrow \theta(t))]_A$ is the A-equivalence class of the term obtained by applying the substitution θ to t and placing the result into $\mathbb{C}$.*

The definition of a single transition of a PRTRT describes how the system state evolves when applying a matching rewrite rule to it:

Definition 6. *Given terms $[u]_A, [v]_A \in Can_{\Sigma,E/A}$, an* E/A-canonical one-step rewrite *from $[u]_A$ to $[v]_A$ is a labelled transition $[u]_A \xrightarrow[\tau]{([\mathbb{C}]_A, r, [\theta]_A, [\rho]_A)} [v]_A$, where:*

[2] Definition 5 also generalizes Definition 3 in [7], to account for the fact that the probability measures can now also depend on variables in the duration term τ_l.

1. $([\mathbb{C}]_A, r, [\theta]_A)$ *is a generalized* R/A*-match for* $[u]_A$ *selected nondeterministically;*
2. $[\rho]_A \in \mathit{CanGSubst}_{E/A}(\vec{p})$ *is a substitution selected with probability* $\pi_r([\theta]_A)([\rho]_A)$*;*
3. *the duration* τ *of the transition is given by* $\theta(\tau_l)$*;*
4. $[v]_A = [\mathbb{C}(\odot \leftarrow t'(\theta(\vec{x}, \vec{y}), \rho(\vec{p})))]_A$ *is the result of the one-step rewrite.*

If $\tau \neq \phi(0)$ *we call* $[u]_A \xrightarrow[\tau]{([\mathbb{C}]_A, r, [\theta]_A, [\rho]_A)} [v]_A$ *an* E/A-canonical one-step tick rewrite*; otherwise we call it an* E/A-canonical one-step instantaneous rewrite*, and we may omit* τ *from the label.*

Intuitively, an E/A-canonical one-step rewrite therefore consists of:

i) nondeterministically choosing a generalized R/A-match $([\mathbb{C}]_A, r, [\theta]_A)$ for $[u]_A$;
ii) selecting a substitution $[\rho]_A$ for $\vec{p}$ with probability $\pi_r([\theta]_A)([\rho]_A)$;
iii) making a (continuous) timed transition by letting time advance by $\tau = \theta(\tau_l)$ time units in the entire system;
iv) making a (discrete) instantaneous transition by applying the substitutions θ and ρ, and putting the result into the context $\mathbb{C}$ that corresponds to the global system state, therefore obtaining a new system state $[v]_A$.

4 Example: A Simple Round Trip Time Protocol

To illustrate our formalism, we specify in an object-oriented way a simple *round trip time* (RTT) protocol that computes the time it takes for a message to go from one node to another, and back, in a network, where the message transmission time follows a probability distribution that depends on the distance between the nodes.

The initiator object `O` starts the protocol by sending an `rttReq` message to its neighbor `O'`, with a time stamp `T` which is the current value of `O`'s local clock (rule `start`). When `O'` receives this message, it immediately sends back a reply to `O` with the original time stamp with probability 3/4 and ignores the request with probability 1/4 (rule `rttResp`). When the initiator `O` receives the reply, it computes its RTT value w.r.t. `O'` by subtracting the original time stamp `T` from its current clock value `T'` (rule `treatResp`). However, if the message takes so long that `T' - T` $\geq$ `maxRTT`, then it is just ignored (rule `ignoreOld`). The initiator uses a retransmission timer to start a new round of the protocol every `maxRTT` time units until it has computed a good RTT value. When the timer expires, `O` sends another RTT request to `O'` (rule `tryAgain`) with probability $1/(\mathtt{N}+1)$, which decreases with the number `N` of unresolved RTT requests of `O`. We represent each node by an object

`<` o `: Node | nbr :` o'`, rtt :` r`, clock :` t`, timer :` ti`, tries :` n `>`

where o is the node's identifier, o' is the neighbor to which o wants to compute its round trip time, r is the value of the round trip time, if computed, or `INF` otherwise, t is the current value of the node's clock, ti is its current timer value, which has the value `INF` if the timer is switched off, and n is the number of unsuccessful attempts that o has made to compute the RTT. Messages have the form `findRtt(`o`)`, which triggers a run of the RTT protocol for node o, `rttReq(`o', o, t`)`, which sends a request from node o to node o' with t the current time stamp of o, and `rttResp(`o, o', t`)`, which sends a reply message from node o' to node o with the original time stamp t. We assume that a function `dist` is defined that computes the distance `dist(`o, o'`)` between two nodes. In the rules `start`, `rttResp` and `tryAgain` we specify the transmission delay of the `rttReq` and `rttResp` messages as a variable `D` which is probabilistically substituted according to a probability distribution `F`(x) that mimics a truncated normal distribution $\mathcal{N}(\mu, \sigma^2)$ [15] with minimum value `minDelay`, and depends on the distance x between o and o', where μ and σ are positive constants representing the average and the standard deviation of the transmission delay, respectively.

The following instantaneous rewrite rules describe our simple RTT protocol. See Appendix A for the full specification of this example.

```
vars O O': Oid . vars T T' D: Time . var N: Nat . var B: Bool . var CF: Configuration .

prl [start] :
    findRtt(O)  < O : Node | clock : T, nbr : O' >
  =>  < O : Node | timer : maxRTT >  dly(rttReq(O', O, T), D)
      with probability D := F(dist(O, O')) .
```

```
prl [rttResp] :
    rttReq(O,O',T)  < O : Node | >
  =>  if B then < O : Node | >  dly(rttResp(O',O,T), D)  else < O : Node | >  fi
      with probability B := ( true  false ) and D := F(dist(O, O')) .
                            ( 3/4   1/4  )

crl [treatResp] :
    rttResp(O,O',T)  < O : Node | clock : T' >
  =>  < O : Node | rtt : T' - T, timer : INF >  if T' - T < maxRTT .

crl [ignoreOld] :
    rttResp(O,O',T)  < O : Node | clock : T' >  =>  < O : Node | >  if T' - T >= maxRTT .

prl [tryAgain] :
    < O : Node | timer : 0, clock : T, nbr : O', tries : N >
  =>  if B then  < O : Node | timer : maxRTT, tries : N + 1 >  dly(rttReq(O,O',T), D)
      else < O : Node | timer : maxRTT >  fi
      with probability B := (   true     false   ) and D := F(dist(O, O')) .
                            ( 1/(N+1)  N/(N+1) )
```

Time elapse is modeled by the tick rule

```
crl [tick] : {CF}  =>  {delta(CF, T)} in time T  if T <= mte(CF) .
```

where `delta` is a frozen function that specifies the effect of time elapse on the system by decreasing the timers and increasing the clock values of each node, as well as decreasing the delays of the messages. The frozen function `mte` gives the maximum amount of time that can elapse before a node must perform an instantaneous transition. More precisely, time cannot advance past the expiration of a timer or the moment when a message arrives. See Appendix A for their formal definition.

It is worth noticing that the number of messages in the state can grow beyond any bound, since: *i)* the message delays could be arbitrarily large (with non-zero probability), and *ii)* the initiator node will periodically send requests until it receives a good RTT value. Therefore, even this simple protocol seems to be beyond the scope of systems that can be defined using automaton-based formalisms.

5 Probability Space over Computation Paths

In this section we define the probability of reaching a given state from another in a certain time in a PRTRT. Our formalism combines probabilistic and nondeterministic behaviors, and therefore we must assign "probabilities" also to the nondeterministic choices to be able to define the probability of reaching a state t_2 from a state t_1 in time τ. This is done by "adversaries," so that the probability of reaching t_2 in time τ is defined relative to a given adversary.

A computation is an infinite sequence of E/A-canonical rewrite steps, with zero-time self-loops from deadlock states:

Definition 7. *A* computation *of a PRTRT is an* infinite *sequence*

$$\Pi = [u_1]_A \xrightarrow[\tau_1]{\alpha_1} [u_2]_A \xrightarrow[\tau_2]{\alpha_2} \cdots \xrightarrow[\tau_{n-1}]{\alpha_{n-1}} [u_n]_A \xrightarrow[\tau_n]{\alpha_n} \cdots, \tag{3}$$

where either each $[u_i]_A \xrightarrow[\tau_i]{\alpha_i} [u_{i+1}]_A$ *is a* E/A*-canonical one-step rewrite, or there exists an index* $n \geq 1$ *such that* $[u_n]_A$ *is a deadlock state, i.e.,* $[u_n]_A$ *cannot be further rewritten using the rules in* R*, in which case* $[u_i]_A \xrightarrow[\tau_i]{\alpha_i} [u_{i+1}]_A$ *is a* E/A*-canonical one-step rewrite for each* $i \leq n-1$*, and* $[u_j]_A = [u_n]_A$*,* $\alpha_j = !$*, and* $\tau_j = 0$ *for each* $j \geq n$*, where '*!*' is a new label.*

Definition 8. *To each computation of the form (3) we associate the* infinite timed computation path $\widehat{\Pi}$ *obtained by removing the labels above the transition arrows, i.e., only keeping the information about timed transitions. A* finite timed computation path *is a prefix* $\widehat{\Pi}_{\text{fin}}$ *of an infinite timed computation path.*

Definition 9. *Given integers* $n, m \geq 2$*, we say that the two finite timed computation paths given by* $\widehat{\Pi}_{\text{fin}} = [u_1]_A \xrightarrow[\tau_1]{} \cdots \xrightarrow[\tau_{n-1}]{} [u_n]_A$ *and* $\widehat{\Pi}'_{\text{fin}} = [u'_1]_A \xrightarrow[\tau'_1]{} \cdots \xrightarrow[\tau'_{m-1}]{} [u'_m]_A$ *are* indistinguishable *if they have the same initial and final states, i.e.,* $[u_1]_A = [u'_1]_A$ *and* $[u_n]_A = [u'_m]_A$*, and their total time duration is the same* $\sum_{i=1}^{n-1} \tau_i = \sum_{j=1}^{m-1} \tau'_j$*. In that case, we write* $\widehat{\Pi}_{\text{fin}} \sim \widehat{\Pi}'_{\text{fin}}$*.*

Notice that $\sim$ is an equivalence relation over the set of all finite timed computation paths of a PRTRT Ψ. In what follows, let $\widehat{\Pi}_{\text{fin}} = [u_1]_A \underset{\tau_1}{\longrightarrow} \ldots \underset{\tau_{n-1}}{\longrightarrow} [u_n]_A$ be a finite timed computation path. Basic cylinder sets are needed to define a probability measure on the set of all computation paths associated with a PRTRT.

Definition 10. *The* basic cylinder set *generated by $\widehat{\Pi}_{\text{fin}}$ is the set $cyl\left(\widehat{\Pi}_{\text{fin}}\right)$ containing all the* infinite *timed computation paths that coincide with $\widehat{\Pi}_{\text{fin}}$ on the first n terms and on the first $n-1$ time values.*

We denote by $\Omega_{[u_1]_A}$ the set of infinite timed computation paths that start with the term $[u_1]_A$. Also, let $\mathcal{B}_{[u_1]_A} \subseteq 2^{\Omega_{[u_1]_A}}$ be the smallest σ-algebra over $\Omega_{[u_1]_A}$ that contains the basic cylinder sets $cyl\left(\widehat{\Pi}_{\text{fin}}\right)$ for all *finite* timed computation paths $\widehat{\Pi}_{\text{fin}}$ that start with $[u_1]_A$. The nondeterministic choices of the generalized R/A-matches $([\mathbb{C}]_A, r, [\theta]_A)$ for $[u_1]_A$ prohibit us from defining a probability measure on $\mathcal{B}_{[u_1]_A}$. To define such a probability measure, all nondeterministic choices must be resolved by means of an *adversary*, which extends the notion of adversary of a probabilistic rewrite theory as follows.

Definition 11. *An* R/A-match adversary *of a PRTRT Ψ is a function $\mathcal{A}$ that maps each finite timed computation path $\widehat{\Pi}_{\text{fin}}$ of Ψ, ending with a term $[u_n]_A$, to a probability measure on the set of generalized R/A-matches for $[u_n]_A$.*

The probability, relative to a given R/A-match adversary $\mathcal{A}$, of performing a given single transition as the next step in a computation is then defined as follows:

Definition 12. *Let $\mathcal{A}$ be an R/A-match adversary of a PRTRT Ψ. The conditional probability that the "non-deadlock" term $[u_n]_A$ rewrites to a term $[u']_A \in Can_{\Sigma,E/A}$ in one step in time τ, provided that $[u_n]_A$ is obtained via $\widehat{\Pi}_{\text{fin}}$, is given by*

$$\mathbb{P}_{\mathcal{A}}\left([u_n]_A \underset{\tau}{\longrightarrow} [u']_A \mid \widehat{\Pi}_{\text{fin}}\right) = \sum \left[\mathcal{A}\left(\widehat{\Pi}_{\text{fin}}\right)([\mathbb{C}]_A, r, [\theta]_A) \cdot \pi_r([\theta]_A)\left(\alpha_r\left([\rho]_A\right)\right)\right],$$

where the sum ranges over all $[u_n]_A \xrightarrow[\tau]{([\mathbb{C}]_A, r, [\theta]_A, [\rho]_A,)} [u']_A$, and α_r is a suitable $\mathcal{F}_r$-cover. If $[u_n]_A$ is a deadlock state then $\mathbb{P}_{\mathcal{A}}\left([u_n]_A \underset{\tau}{\longrightarrow} [u']_A \mid \widehat{\Pi}_{\text{fin}}\right) = 1$ if and only if $[u']_A = [u_n]_A$ and $\tau = 0$, and is 0 otherwise.

Given an adversary $\mathcal{A}$, the following definition then states the probability of having a path of the form $\widehat{\Pi}_{\text{fin}} = [u_1]_A \underset{\tau_1}{\longrightarrow} \ldots \underset{\tau_{n-1}}{\longrightarrow} [u_n]_A$:

Definition 13. *The probability associated with a basic cylinder set generated by the finite timed computation path Π_{fin} is given by:*

$$\mathbb{P}_{\mathcal{A}}\left(cyl\left(\widehat{\Pi}_{\text{fin}}\right)\right) = \prod_{i=1}^{n-1} \mathbb{P}_{\mathcal{A}}\left([u_i]_A \underset{\tau_i}{\longrightarrow} [u_{i+1}]_A \;\middle|\; [u_1]_A \underset{\tau_1}{\longrightarrow} \ldots \underset{\tau_{i-1}}{\longrightarrow} [u_i]_A\right).$$

The probabilities associated with the basic cylinder sets give rise to a unique probability measure on the σ-algebra $\mathcal{B}_{[u_1]_A}$ over the set of all infinite timed computation paths, starting at $[u_1]_A$, of a PRTRT. In the following result, we prove that the probability of reaching a state $[u']_A$ from a state $[u]_A$ in time τ, w.r.t. given adversaries, is just the sum of the probabilities of having paths $\widehat{\Pi}_{\text{fin}}$ whose total duration is τ and where there is no earlier occurrence of $[u']_A$ reached in time τ. We use the notation $[u]_A \overset{*}{\underset{\tau}{\rightarrow}} [u']_A$ to express the fact that there exists a finite timed computation path such that its first state is $[u]_A$, its last state is $[u']_A$, and its total duration is τ.

Proposition 1. *The probability of reaching a state $[u']_A$ in time τ from state $[u]_A$ is given by*

$$\mathbb{P}_{\mathcal{A}}\left([u]_A \overset{*}{\underset{\tau}{\rightarrow}} [u']_A\right) = \sum_{\widehat{\Pi}_{\text{fin}}} \mathbb{P}_{\mathcal{A}}\left(cyl\left(\widehat{\Pi}_{\text{fin}}\right)\right) \tag{4}$$

where $\widehat{\Pi}_{\text{fin}}$ ranges over all finite timed computation paths $[u]_A = [u_1]_A \underset{\tau_1}{\longrightarrow} \ldots \underset{\tau_{k-1}}{\longrightarrow} [u_k]_A = [u']_A$ with $\sum_{i=1}^{k-1} \tau_i = \tau$, and there is no $j < k$ with $[u_j]_A = [u']_A$ and $\sum_{i=1}^{j-1} \tau_i = \tau$.

Proof. Let P and P' be two finite timed computation paths that satisfy the same conditions as the indices $\widehat{\Pi}_{\text{fin}}$ of the sum in (4). Then it cannot be the case that P is a prefix of P', or vice-versa. However, $cyl(P) \cap cyl(P') \neq \emptyset$ holds if and only if P is a prefix of P'. Therefore, for any indices P and P' of the sum in (4) we have that $cyl(P) \cap cyl(P') = \emptyset$, and the formula follows from the inclusion-exclusion principle and the addition law of probability.

6 The Expressive Power of PRTRTs

In this section we show the expressiveness of PRTRTs—and its suitability as a unifying semantic framework for probabilistic real-time systems in which different models of such systems can be represented—by explaining how a range of models of probabilistic real-time systems can naturally be seen as PRTRTs. We also give details about the corresponding mappings and their correctness proofs.

Since probabilistic rewrite theories are a proper subclass of PRTRTs, any model that can be expressed as a probabilistic rewrite theory can also be represented as a PRTRT. In [18] mappings are provided from probabilistic nondeterministic systems, generalized semi-Markov processes, and continuous-time Markov chains into probabilistic rewrite theories. That paper also claims that the same method can be used for representing the PEPA [14] language and various Petri net formalisms, such as stochastic reward nets, generalized stochastic Petri nets [3], and stochastic Petri nets with generally distributed firing times, as probabilistic rewrite theories.

6.1 Probabilistic Timed Automata

The *probabilistic timed automaton* (PTA) model [32] combines nondeterministic and probabilistic behaviors, and extends timed automata [5] by allowing a probabilistic choice of both the *next state* and the *set of clocks* to be reset in a "transition." PTA are also supported by recent versions of the probabilistic model checker PRISM [19].

A *clock* is a variable ranging over the real numbers that increases its value according to the elapsed time. A *zone* of a set of clocks $\mathcal{X}$ is a convex subset of $\mathbb{R}^{|\mathcal{X}|}$ defined by a conjunction of constraints over $\mathcal{X}$. Denote by $Z_{\mathcal{X}}$ the set of all zones of a set of clocks $\mathcal{X}$.

Definition 14. *A PTA [32] is a tuple* $(S, s_0, \mathcal{X}, \text{inv}, \text{prob}, \{\tau_s\}_{s \in S})$*, where:*

- S *is a finite set of* states *with* $s_0 \in S$ *the* start *state;*
- $\mathcal{X}$ *is a finite set of* clocks*;*
- $\text{inv} : S \to Z_{\mathcal{X}}$ *is a function that assigns an* invariant condition *to each state, such that* $0 \in \text{inv}(s_0)$*, i.e., the invariant of the initial state is satisfied at the beginning of a run of the PTA, when all clocks in* $\mathcal{X}$ *are set to* 0*;*
- $\text{prob} : S \to \mathcal{P}(PMF\,(S \times \mathcal{P}(\mathcal{X})))$ *is a function that* assigns a set of probability distributions *on* $S \times \mathcal{P}(\mathcal{X})$ *to each state;*
- $\{\tau_s\}_{s \in S}$ *is a family of functions where, for each* $s \in S$*,* $\tau_s : \text{prob}(s) \to Z_{\mathcal{X}}$ *assigns an* enabling condition *to each probability distribution* $p \in \text{prob}(s)$*.*

Following [20, 32], we assume that PTA are subject to the following conditions:

- If the PTA is in some state s and time cannot elapse without violating the invariant condition of s, we assume that there always exists a probability distribution $p \in \text{prob}(s)$ whose enabling condition $\tau_s(p)$ is satisfied.
- The enabling condition $\tau_s(p)$ of any probability distribution $p \in \text{prob}(s)$ implies the invariant of all possible successor states s' with $p(s', X) > 0$, *after* the clocks in X are reset. This is also known as the "admissible targets" assumption.

A PTA in state s may *nondeterministically* select any enabled probability distribution p in $\text{prob}(s)$. The probability that the automaton then makes a transition to state s' and resets all the clocks $X \subseteq \mathcal{X}$ to 0 is $p(s', X)$.

Example 2. Figure 1 shows a PTA that starts in state s_0 with its single clock x initialized to 0. The automaton may wait in state s_0 for at most 8 time units, since $\text{inv}(s_0) = [0, 8]$. When $x \in [5, 7]$ in state s_0 the automaton can nondeterministically choose between the two distributions $\pi_1 = \begin{pmatrix} (s_1, \emptyset) & (s_2, \emptyset) \\ 0.3 & 0.7 \end{pmatrix}$ and $\pi_2 = \begin{pmatrix} (s_2, \emptyset) \\ 1 \end{pmatrix}$; when $x \in [3, 8] \setminus [5, 7]$ it can only take π_2. As soon as a probability distributions is chosen,

a probabilistic choice is made, and the corresponding instantaneous transition is performed. The PTA is allowed to make a probabilistic transition according to the distribution π_1 when $x \in [5,7]$. Likewise, the probabilistic choice from s_1 is made by sampling from the probability distribution $\pi_3 = \begin{pmatrix} (s_1,\emptyset) & (s_2,\{x\}) \\ 0.5 & 0.5 \end{pmatrix}$; if the automaton makes a transition to state s_2, which happens with probability 0.5, then x is reset to 0.

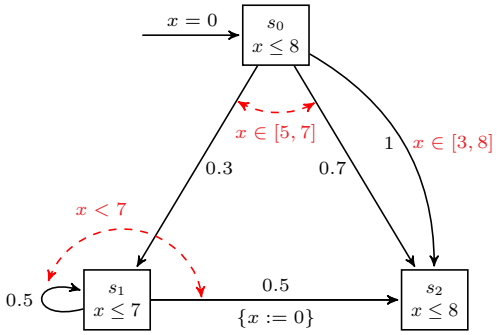

Fig. 1. A probabilistic timed automaton with one clock.

Translation into PRTRT. A PTA $A = (S, s_0, \mathcal{X}, \text{inv}, \text{prob}, \langle \tau_s \rangle_{s \in S})$, where we assume without loss of generality that $\mathcal{X}$ is a set of clocks $\{x_1, \ldots, x_n\}$, is represented as a PRTRT $\Psi_{PTA}(A) = (\Sigma, \emptyset, E, L, R, \pi, \phi, \tau)$ as follows. The MEL theory (Σ, E) contains an equational axiomatization of a computable subfield of $\mathbb{R}$, e.g., the set of rationals or the algebraic real numbers, that defines the sort `Time` denoting the time domain. The signature Σ contains a sort `PTAState` with a constant s for each state $s \in S$, and a $(|\mathcal{X}|+1)$-ary operator $\_\,,\_\,,\ldots,\_ : \texttt{PTAState Time} \ldots \texttt{Time} \to \texttt{System}$. A "timed state" of the PTA is represented as a term $\{s, r_1, r_2, \ldots, r_n\}$, with s the current state and with r_i denoting the current value of clock x_i. Then, to each state $s \in S$ and each probability distribution $\pi : S \times \mathcal{P}(\mathcal{X}) \to [0,1]$ in $\text{prob}(s)$, we associate an instantaneous probabilistic rewrite rule $r \in R$ of the form

$$\texttt{crl [}\pi\texttt{]: } \{s, y_1, \ldots, y_n\} \texttt{ => } \sigma \texttt{ if } (y_1, \ldots, y_n) \in \tau_s(\pi) \texttt{ with probability } \sigma := \Gamma_s(\pi) \tag{5}$$

where σ and the y_i are variables, and $\Gamma_s(\pi) : \mathcal{F}_r \to [0,1]$ is a probability measure on $\mathit{CanGSubst}_{E/A}(\sigma)$ given by

$$\Gamma_s(\pi)(\sigma \mapsto \{s', t_1, \ldots, t_n\}) = \pi(s', X) \tag{6}$$

for all $s' \in S$ and all $X \subseteq \mathcal{X}$, where t_j equals 0 if $x_j \in X$ and y_j otherwise. To model time elapse, we also add to R a tick rewrite rule

$$\texttt{crl [tick}\,s\texttt{]: } \{s, y_1, \ldots, y_n\} \texttt{ => } \{s, y_1 + y, \ldots, y_n + y\} \texttt{ in time } y \texttt{ if } (y_1 + y, \ldots, y_n + y) \in \text{inv}(s) \tag{7}$$

for each $s \in S$, where y is a variable. Since $\tau_s(\pi)$ and $\text{inv}(s)$ are zones defined by conjunctions of inequality constraints over the clock values, the two set memberships in the above rules, i.e., $(y_1, y_2, \ldots, y_n) \in \tau_s(\pi)$ and $(y_1 + y, y_2 + y, \ldots, y_n + y) \in \text{inv}(s)$, are translated into standard inequalities in $\Psi_{PTA}(A)$, as shown in the example below.

Example 3. The probabilistic timed automaton in Fig. 1 is represented by a PRTRT containing the following set of conditional tick rules

```
crl [tick s0]: {s0, x} => {s0, x + y} in time y if x + y <= 8 .
crl [tick s1]: {s1, x} => {s1, x + y} in time y if x + y <= 7 .
crl [tick s2]: {s2, x} => {s2, x + y} in time y if x + y <= 8 .
```

as well as the following instantaneous probabilistic rewrite rules

$$\texttt{crl [}\pi_1\texttt{]: } \{s_0, x\} \texttt{ => } \sigma \texttt{ if } x \texttt{ >= 5 and } x \texttt{ <= 7 with probability } \sigma := \begin{pmatrix} \{s_1, x\} & \{s_2, x\} \\ 0.3 & 0.7 \end{pmatrix}.$$

$$\texttt{crl [}\pi_2\texttt{]: } \{s_0, x\} \texttt{ => } \sigma \texttt{ if } x \texttt{ >= 3 and } x \texttt{ <= 8 with probability } \sigma := \begin{pmatrix} \{s_2, x\} \\ 1.0 \end{pmatrix}.$$

$$\texttt{crl [}\pi_3\texttt{]: } \{s_1, x\} \texttt{ => } \sigma \texttt{ if } x \texttt{ < 7 with probability } \sigma := \begin{pmatrix} \{s_1, x\} & \{s_2, 0\} \\ 0.5 & 0.5 \end{pmatrix}.$$

where σ, x, and y are variables and the initial state is given by the term $\{s_0, 0\}$.

Correspondence Theorem. Given a probabilistic timed automaton $A = (S, s_0, \mathcal{X}, \text{inv}, \text{prob}, \{\tau_s\}_{s\in S})$, its *timed states* are pairs $(s, v) \in S \times \mathbb{R}^{\mathcal{X}}$, with $s \in S$ a state and $v : \mathcal{X} \to [0, \infty)$ a clock valuation, such that $v \in \text{inv}(s)$. In order to be able to prove (Theorem 1) that the correspondence between PTA and their PRTRT representation also preserves the probabilistic choices made along their execution, we provide an extended definition of the runs of a PTA that includes this information:

Definition 15. *A* run ρ *of a PTA with initial state* s_0 *is an infinite labeled sequence of timed states, of the form*

$$\rho : \ (s_0, 0) \xrightarrow[t_1]{p_1} (s_1, v_1) \xrightarrow[t_2]{p_2} (s_2, v_2) \xrightarrow[t_3]{p_3} \dots \tag{8}$$

where, for all $i \geq 1$*:*

1. $t_i \in \mathbb{R}$ *gives the current time, right before the PTA makes an instantaneous, probabilistic transition to* s_i*;*
2. *the probability distribution* $p_i \in \text{prob}(s_i)$ *satisfies the enabling condition* $v_{i-1} + (t_i - t_{i-1}) \in \tau_{s_{i-1}}(p_i)$*, with* $t_0 = 0$*;*
3. *the invariant in state* s_{i-1} *must hold before making a probabilistic transition to* s_i *via the probability distribution* p_i*, i.e., it must be the case that* $v_{i-1} + (t_i - t_{i-1}) \in \text{inv}(s_{i-1})$*;*
4. *the timed state* (s_i, v_i) *is obtained from* (s_{i-1}, v_{i-1}) *by sampling the pair* $(s_i, \widetilde{X}_i) \in S \times \mathcal{P}(\mathcal{X})$ *from the probability distribution* p_i*, and setting* $v_i(x) = 0$ *for all* $x \in \widetilde{X}_i$*, as well as* $v_i(y) = v_{i-1}(y) + (t_i - t_{i-1})$ *for all* $y \notin \widetilde{X}_i$*.*

We let $runs(A)$ *denote the set of all runs of a PTA* A*.*

Notice that, for all $i \geq 1$, the invariant condition in state s_{i-1} always holds throughout the timed transition from time instant v_{i-1} until $v_{i-1} + (t_i - t_{i-1})$, and this is due to:

- the "admissible targets" assumption in the definition of a PTA, together with $0 \in \text{inv}(s_0)$, which ensure that $v_j \in \text{inv}(s_j)$ for all $j \geq 0$, and in particular that $v_{i-1} \in \text{inv}(s_{i-1})$;
- the condition $v_{i-1} + (t_i - t_{i-1}) \in \text{inv}(s_{i-1})$ in the definition of a PTA run;
- the fact that the set of all clock valuations, from v_{i-1} to $v_{i-1} + (t_i - t_{i-1})$ is a line segment in $\mathbb{R}^n$, parallel to the diagonal $\{(x, \dots, x) \mid x \in \mathbb{R}\}$, since all clocks in $\mathcal{X}$ advance at the same time, and at the same rate;
- most importantly, the convexity of the set $\text{inv}(s_{i-1})$.

We now define elementary computation steps in $\Psi_{PTA}(A)$, which correspond to single steps in a run of a PTA.

Definition 16. *A* PTA-step φ *in* $\Psi_{PTA}(A)$ *is a finite sequence of* E/A*-canonical one-step* tick *rewrites, followed by an* E/A*-canonical one-step* instantaneous *rewrite.*

Notice that, based on our translation of PTA into PRTRTs, all PTA-steps must be of the following form

$$\begin{aligned}\varphi_i : \ \left[\{\sigma_i, y_i^1, \dots, y_i^n\}\right]_A \xrightarrow[r_i^1]{\beta_i^1} \left[\{\sigma_i, y_i^1 + r_i^1, \dots, y_i^n + r_i^1\}\right]_A \xrightarrow[r_i^2]{\beta_i^2} \\ \xrightarrow[r_i^2]{\beta_i^2} \left[\{\sigma_i, y_i^1 + (r_i^1 + r_i^2), \dots, y_i^n + (r_i^1 + r_i^2)\}\right]_A \xrightarrow[r_i^3]{\beta_i^3} \dots \\ \xrightarrow[r_i^{k_i}]{\beta_i^{k_i}} \left[\{\sigma_i, y_i^1 + \left(\textstyle\sum_{j=1}^{k_i} r_i^j\right), \dots, y_i^n + \left(\textstyle\sum_{j=1}^{k_i} r_i^j\right)\}\right]_A \xrightarrow[\phi(0)]{\beta_i} \left[\{\sigma_{i+1}, y_{i+1}^1, \dots, y_{i+1}^n\}\right]_A ,\end{aligned} \tag{9}$$

where σ_i, σ_{i+1} are terms of sort `PTAState`, $y_i^1, \dots, y_i^n$ and $y_{i+1}^1, \dots, y_{i+1}^n$ are terms of sort `Time`, $k_i \geq 0$ is the number of tick rewrites in φ_i with $r_i^1, \dots, r_i^{k_i}$ their associated durations and $\beta_i^1, \dots, \beta_i^{k_i}$ the corresponding labels of the E/A-canonical one-step tick rewrites, which contain a tick rule of the form (7) labeled `[tick` σ_i`]`, and β_i is the label of the last E/A-canonical one-step instantaneous rewrite in φ_i, which contains an instantaneous probabilistic rewrite rule of the form (5) labeled by `[`π_i`]`, where $\pi_i \in \text{prob}(\sigma_i)$ is a probability distribution available in σ_i.

Definition 17. *A PTA-step which contains a single* E/A*-canonical one-step tick rewrite is called a* minimal *PTA-step.*

Based on our translation, it follows that any PTA-step of the form (9) with $k_i = 1$ is minimal. The following definition introduces an equivalence relation over PTA-steps which allows us to only focus on minimal PTA-steps when reasoning about arbitrary PTA-steps in $\Psi_{PTA}(A)$, since they contain all the necessary information and they are *indistinguishable* from other non-minimal, equivalent PTA-steps.

Definition 18. *Two PTA-steps*

$$\varphi: \ [u_0]_A \xrightarrow[r_1]{\beta_1} [u_1]_A \xrightarrow[r_2]{\beta_2} \dots \xrightarrow[r_n]{\beta_n} [u_n]_A \xrightarrow[\phi(0)]{\beta} [v]_A$$

and

$$\varphi': \ [u'_0]_A \xrightarrow[r'_1]{\beta'_1} [u'_1]_A \xrightarrow[r'_2]{\beta'_2} \dots \xrightarrow[r'_m]{\beta'_m} [u'_m]_A \xrightarrow[\phi(0)]{\beta'} [v']_A$$

are indistinguishable*, and we write $\varphi \sim \varphi'$ if their associated computation paths $[u_0]_A \xrightarrow[r_1]{} \dots \xrightarrow[r_n]{} [u_n]_A$ and $[u'_0]_A \xrightarrow[r'_1]{} \dots \xrightarrow[r'_m]{} [u'_m]_A$ are indistinguishable (see Definition 9), and $\beta = \beta'$, which also implies $[v]_A = [v']_A$.*

It follows easily that the indistinguishability relation $\sim$ over the set of all PTA-steps in $\Psi_{PTA}(A)$ is an equivalence relation.

Lemma 1. *Let φ be a PTA-step in $\Psi_{PTA}(A)$. Then there exists a unique minimal PTA-step φ^{min} which is indistinguishable from φ.*

Proof. Let φ_i be an arbitrary PTA-step in $\Psi_{PTA}(A)$, of the form (9). We use the fact that any application of the tick rewrite rule (7) in φ_i only affects the clock terms $y_i^1, \dots, y_i^n$, without changing the current state σ_i. Therefore, we can replace a series of timed transitions with a single one, whose duration is given by the sum of durations in the original series. This allows us to pick the minimal PTA-step

$$\varphi_i^{min} \ : \quad \left[\{\sigma_i, y_i^1, \dots, y_i^n\}\right]_A \xrightarrow[r_i]{\alpha_i} \left[\{\sigma_i, y_i^1 + r_i, \dots, y_i^n + r_i\}\right]_A \xrightarrow[\phi(0)]{\beta_i} \left[\{\sigma_{i+1}, y_{i+1}^1, \dots, y_{i+1}^n\}\right]_A \quad (10)$$

which is indistinguishable from φ_i, where the label α_i of the E/A-canonical one-step tick rewrite contains the substitution $\theta = \left\{y_1 \mapsto y_i^1, \dots, y_n \mapsto y_i^n, y \mapsto \sum_{j=1}^{k_i} r_i^j\right\}$ for matching against the tick rewrite rule (7), and $r_i = \sum_{j=1}^{k_i} r_i^j$ is the total duration of the E/A-canonical one-step tick rewrites in φ_i. Since φ_i^{min} is completely determined by the terms and labels of φ_i, it follows that φ_i^{min} is also unique. □

Notice that the mapping $\varphi \mapsto \varphi^{min}$ is surjective, but it is not injective since there exist several PTA-steps, with the same total duration, which are mapped into the same corresponding minimal PTA-step. We next define PTA-computations as infinite sequences of PTA-steps in $\Psi_{PTA}(A)$.

Definition 19. *A* PTA-computation *of the PRTRT representation $\Psi_{PTA}(A)$ of a PTA A is an infinite sequence $\{\varphi_i\}_{i\geq 0}$ of PTA-steps, where φ_i is of the form (9) for all $i \geq 0$, and $y_0^1 = \dots = y_0^n = 0$. We denote by PTA–$\mathcal{C}(\Psi_{PTA}(A))$ the set of all PTA-computations of the probabilistic real-time rewrite theory $\Psi_{PTA}(A)$.*

Similar to the case of PTA-steps, we can define minimal PTA-computations as follows.

Definition 20. *A* minimal *PTA-computation is one whose terms are all minimal PTA-steps. We denote by PTA–$\mathcal{C}_{min}(\Psi_{PTA}(A))$ the set of all minimal PTA-computations of $\Psi_{PTA}(A)$.*

The indistinguishability relation generalizes from PTA-steps to PTA-computations in a natural way.

Definition 21. *Two PTA-computations $\{\varphi_i\}_{i\geq 0}$ and $\{\varphi'_i\}_{i\geq 0}$ are* indistinguishable *if and only if their corresponding terms are indistinguishable PTA-steps, i.e., $\varphi_i \sim \varphi'_i$ for all $i \geq 0$. In that case, we write $\{\varphi_i\}_{i\geq 0} \sim \{\varphi'_i\}_{i\geq 0}$.*

Notice that $\sim$ is an equivalence relation over the set of all PTA-computations PTA–$\mathcal{C}(\Psi_{PTA}(A))$, and we write $[\rho]_\sim$ for the equivalence class of a PTA-computation ρ. As in the case of PTA-steps, given a PTA-computation $\{\varphi_i\}_{i\geq 0}$, there exists exactly one minimal PTA-computation $\{\varphi_i^{min}\}_{i\geq 0}$, also denoted $\{\varphi_i\}_{i\geq 0}^{min}$, which is indistinguishable from $\{\varphi_i\}_{i\geq 0}$. Furthermore, the mapping $\{\varphi_i\}_{i\geq 0} \mapsto \{\varphi_i\}_{i\geq 0}^{min}$

is surjective, but it is not injective since there exist several PTA-computations, whose PTA-steps have the same corresponding total duration, and which are mapped into the same minimal PTA-computation.

Given two PTA-computations $\{\varphi_i\}_{i\geq 0}$ and $\{\varphi'_i\}_{i\geq 0}$, they are indistinguishable if and only if they have the same corresponding minimal PTA-computation $\{\varphi_i\}_{i\geq 0}^{min} = \{\varphi'_i\}_{i\geq 0}^{min}$. This follows from the transitivity of the indistinguishability relation and the fact that indistinguishable PTA-computations have the same total duration. Therefore, each minimal PTA-computation uniquely determines an equivalence class of indistinguishable PTA-computations, and the quotient set $PTA\text{–}\mathcal{C}(\Psi_{PTA}(A))/_\sim$ can be indexed by the elements of $PTA\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A))$:

$$PTA\text{–}\mathcal{C}(\Psi_{PTA}(A))/_\sim = \{\, [\rho]_\sim \mid \rho \in PTA\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A)) \,\}. \tag{11}$$

The set $PTA\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A))$ therefore contains the "essential" PTA-computations of $\Psi_{PTA}(A)$, i.e., all other PTA-computations are indistinguishable from the ones in this set. The next lemma says that, to each run of a PTA A there corresponds a unique *minimal* PTA-computation of $\Psi_{PTA}(A)$, which implies that there exists a *functional relation* from the set $runs(A)$ of all runs of A, to the set $PTA\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A))$ of all minimal PTA-computations of $\Psi_{PTA}(A)$.

Lemma 2. *Let A be a probabilistic timed automaton. To each run $\rho \in runs(A)$ of A the form (8) there corresponds a unique minimal PTA-computation $\widetilde{\rho} = \{\widetilde{\varphi}_i\}_{i\geq 0} \in PTA\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A))$, with the term $\widetilde{\varphi}_i$ of the form* (10) *such that for all $i \geq 0$:*

- *the term σ_i coincides with the state s_i of the PTA, i.e., $\sigma_i = s_i$;*
- *the term y_i^j of sort* `Time` *coincides with the value of clock x_j in the clock valuation v_i of the PTA, i.e., $y_i^j = v_i(x_j)$ for all $j \in \{1, \ldots, n\}$;*
- *the duration term r_i is given by the time difference $t_{i+1} - t_i$, i.e., the PTA A and the PRTRT $\Psi_{PTA}(A)$ take the instantaneous, probabilistic transitions at the same time;*
- *the label β_i of the E/A-canonical one-step instantaneous rewrite is given by $\beta_i = ([\mathbb{C}]_A, r, [\theta]_A, [\rho]_A)$ with r a probabilistic rewrite rule of the form (5) labeled by the probability distribution $\pi_i = p_i$.*

Proof. Let $\rho \in runs(A)$ be an arbitrary run of the PTA A. The PTA-step $\widetilde{\varphi}_i$ of the minimal PTA-computation $\{\widetilde{\varphi}_i\}_{i\geq 0}$ corresponding to ρ, which satisfies the properties listed in the lemma, must be of the following form

$$\widetilde{\varphi}_i :\ [\{s_i, v_i(x_1), \ldots, v_i(x_n)\}]_A \xrightarrow[t_{i+1}-t_i]{\alpha_i} [\{s_i, v_i(x_1) + (t_{i+1} - t_i), \ldots, v_i(x_n) + (t_{i+1} - t_i)\}]_A \xrightarrow[\phi(0)]{\beta_i} [\{s_{i+1}, v_{i+1}(x_1), \ldots, v_{i+1}(x_n)\}]_A\,, \tag{12}$$

with the instantaneous probabilistic rewrite rule in β_i of the form (5) and labeled by the probability distribution p_i, for all $i \geq 0$. We prove that such a minimal PTA-computation can actually be obtained in $\Psi_{PTA}(A)$ by using generalized R/A-matches and applying E/A-canonical one-step rewrites. We proceed by induction on the number of PTA-steps of $\{\widetilde{\varphi}_i\}_{i\geq 0}$. The first PTA-step is of the form:

$$\widetilde{\varphi}_0 :\ [\{s_0, 0, \ldots, 0\}]_A \xrightarrow[t_1]{\alpha_1} [\{s_0, t_1, \ldots, t_1\}]_A \xrightarrow[\phi(0)]{\beta_1} [\{s_1, v_1(x_1), \ldots, v_1(x_n)\}]_A\,.$$

We take each E/A-canonical one-step rewrite of this PTA-step separately, and show that it can be obtained in $\Psi_{PTA}(A)$.

i) The E/A-canonical one-step tick rewrite $[\{s_0, 0, \ldots, 0\}]_A \xrightarrow[t_1]{\alpha_1} [\{s_0, t_1, \ldots, t_1\}]_A$ is obtained as follows. First the generalized R/A-match $([\mathbb{C}]_A, r, [\theta]_A)$ is selected for the term $\{s_0, 0, \ldots, 0\}$, where $\mathbb{C} = \odot$ is the context, r is a conditional tick rule of the form (7) labeled by $[\mathtt{tick}_{s_0}]$, the substitution θ is given by $\{y_1 \mapsto 0, \ldots, y_n \mapsto 0, y \mapsto t_1\}$, and $\alpha_1 = ([\mathbb{C}]_A, r, [\theta]_A, \emptyset)$. The fact that the substitution θ satisfies the rule's condition $(y_1 + y, \ldots, y_n + y) \in \mathrm{inv}(s_0)$ is therefore equivalent to whether $(t_1, \ldots, t_1) \in \mathrm{inv}(s_0)$, which is true since the PTA run ρ satisfies $v_0 + t_1 \in \mathrm{inv}(s_0)$ (Definition 15, assumption 3). Hence, by applying the tick rule (7), the term $[\{s_0, t_1, \ldots, t_1\}]_A$ is obtained.

ii) The E/A-canonical one-step instantaneous rewrite

$$[\{s_0, t_1, \ldots, t_1\}]_A \xrightarrow[\phi(0)]{\beta_1} [\{s_1, v_1(x_1), \ldots, v_1(x_n)\}]_A$$

can also be obtained in $\Psi_{PTA}(A)$ as follows. The generalized R/A-match $([\mathbb{C}']_A, r', [\theta']_A)$ is first selected for the term $\{\!\{s_0, t_1, \ldots, t_1\}\!\}$, with $\mathbb{C} = \odot$, r a conditional probabilistic rewrite rule of the form (5) labeled by $[p_1]$ and with $s = s_0$, and $\theta = \{y_1 \mapsto t_1, \ldots, y_n \mapsto t_1\}$. The fact that the substitution θ satisfies the rule's condition $(y_1, \ldots, y_n) \in \tau_{s_0}(p_1)$, is therefore equivalent to whether $(t_1, \ldots, t_1) \in \tau_{s_0}(p_1)$, and this is true since the PTA run ρ satisfies $v_0 + t_1 \in \tau_{s_0}(p_1)$ (Definition 15, assumption 2). The substitution $\rho = \{\sigma \mapsto \{\!\{s_1, q_1, \ldots, q_n\}\!\}\}$ contained in β_1 is selected with probability $p_1(s_1, X)$, for a set of clocks to reset $X \subseteq \mathcal{X}$, where $q_j = 0$ if $x_j \in X$ and $q_j = t_1$ otherwise, for all $j \in \{1, \ldots, n\}$. Since the timed state (s_1, v_1) is obtained in the PTA run ρ with nonzero probability $p_1(s_1, \widetilde{X}_1)$ (Definition 14, assumption 4), there exists an E/A-canonical one-step rewrite which selects the substitution ρ such that $q_j = v_1(x_j)$, for all $j \in \{1, \ldots, n\}$; the result of this one-step rewrite is the term $[\{\!\{s_1, v_1(x_1), \ldots, v_1(x_n)\}\!\}]_A$.

Now let $i \geq 1$ be arbitrary and assume, by the induction hypothesis, that the minimal PTA-computation prefix $\{\widetilde{\varphi}_k\}_{k=0}^{i-1}$ corresponding to the i steps prefix of the PTA run ρ

$$(s_0, 0) \xrightarrow[t_1]{p_1} (s_1, v_1) \xrightarrow[t_2]{p_2} \ldots \xrightarrow[t_i]{p_i} (s_i, v_i),$$

can be obtained in $\Psi_{PTA}(A)$. Under this assumption, we must prove that the PTA-step

$$\widetilde{\varphi}_i : \ [\{\!\{s_i, v_i(x_1), \ldots, v_i(x_n)\}\!\}]_A \xrightarrow[t_{i+1}-t_i]{\alpha_i} [\{\!\{s_i, v_i(x_1) + (t_{i+1} - t_i), \ldots, v_i(x_n) + (t_{i+1} - t_i)\}\!\}]_A \xrightarrow[\phi(0)]{\beta_i} [\{\!\{s_{i+1}, v_{i+1}(x_1), \ldots, v_{i+1}(x_n)\}\!\}]_A$$

corresponding to the probabilistic timed transition $(s_i, v_i) \xrightarrow[t_{i+1}]{p_{i+1}} (s_{i+1}, v_{i+1})$ in the PTA run ρ, can also be obtained in $\Psi_{PTA}(A)$. We again consider each E/A-canonical one-step rewrite of this PTA-step separately.

i) The E/A-canonical one-step tick rewrite

$$[\{\!\{s_i, v_i(x_1), \ldots, v_i(x_n)\}\!\}]_A \xrightarrow[t_{i+1}-t_i]{\alpha_i} [\{\!\{s_i, v_i(x_1) + (t_{i+1} - t_i), \ldots, v_i(x_n) + (t_{i+1} - t_i)\}\!\}]_A$$

is obtained in the following manner. The generalized R/A-match $([\mathbb{C}]_A, r, [\theta]_A)$ is selected for the term $\{\!\{s_i, v_i(x_1), \ldots, v_i(x_n)\}\!\}$, where $\mathbb{C} = \odot$ is the context, r is a conditional tick rule of the form (7) labeled by $[\texttt{tick}_{s_i}]$, $\theta = \{y_1 \mapsto v_i(x_1), \ldots, y_n \mapsto v_i(x_n), y \mapsto (t_{i+1} - t_i)\}$, and $\alpha_i = ([\mathbb{C}]_A, r, [\theta]_A, \emptyset)$. The fact that the substitution θ satisfies the rule's condition $(y_1 + y, \ldots, y_n + y) \in \mathrm{inv}(s_i)$ is equivalent to whether

$$(v_i(x_1) + (t_{i+1} - t_i), \ldots, v_i(x_n) + (t_{i+1} - t_i)) \in \mathrm{inv}(s_i),$$

which is true since the PTA run ρ satisfies $v_i + (t_{i+1} - t_i) \in \mathrm{inv}(s_i)$ (Definition 15, assumption 3). By applying the tick rule (7), the term $[\{\!\{s_i, v_i(x_1) + (t_{i+1} - t_i), \ldots, v_i(x_n) + (t_{i+1} - t_i)\}\!\}]_A$ is hence obtained.

ii) The E/A-canonical one-step instantaneous rewrite

$$[\{\!\{s_i, v_i(x_1) + (t_{i+1} - t_i), \ldots, v_i(x_n) + (t_{i+1} - t_i)\}\!\}]_A \xrightarrow[\phi(0)]{\beta_i} [\{\!\{s_{i+1}, v_{i+1}(x_1), \ldots, v_{i+1}(x_n)\}\!\}]_A$$

is also valid in $\Psi_{PTA}(A)$, and can be obtained as follows. The generalized R/A-match $([\mathbb{C}']_A, r', [\theta']_A)$ is picked for the term $[\{\!\{s_i, v_i(x_1) + (t_{i+1} - t_i), \ldots, v_i(x_n) + (t_{i+1} - t_i)\}\!\}]_A$ with $\mathbb{C} = \odot$, r a conditional probabilistic rewrite rule of the form (5) labeled by $[p_{i+1}]$ and with $s = s_i$, and:

$$\theta = \{y_1 \mapsto v_i(x_1) + (t_{i+1} - t_i), \ldots, y_n \mapsto v_i(x_n) + (t_{i+1} - t_i)\}.$$

The fact that the substitution θ satisfies the rule's condition $(y_1, \ldots, y_n) \in \tau_{s_i}(p_{i+1})$ is equivalent to

$$(v_i(x_1) + (t_{i+1} - t_i), \ldots, v_i(x_n) + (t_{i+1} - t_i)) \in \tau_{s_i}(p_{i+1}),$$

and this is true since the PTA run ρ satisfies $v_i + (t_{i+1} - t_i) \in \tau_{s_i}(p_{i+1})$ for all $i \geq 0$ (Definition 15, assumption 2). Finally, the term $[\{\!\{s_{i+1}, v_{i+1}(x_1), \ldots, v_{i+1}(x_n)\}\!\}]_A$ is obtained as a result of the E/A-canonical one-step instantaneous rewrite, where the substitution $\rho = \{\sigma \mapsto \{\!\{s_{i+1}, q_1, \ldots, q_n\}\!\}\}$

is selected with probability $p_{i+1}(s_{i+1}, X)$, for a set of clocks to reset $X \subseteq \mathcal{X}$, where $q_j = 0$ if $x_j \in X$ and $q_j = v_i(x_j)$ otherwise, for all $j \in \{1, \ldots, n\}$. Since the timed state (s_{i+1}, v_{i+1}) has been obtained in the PTA run ρ with nonzero probability $p_{i+1}(s_{i+1}, \widetilde{X}_{i+1})$ (Definition 14, assumption 4), then there exists an E/A-canonical one-step rewrite which selects the substitution ρ such that $q_j = v_{i+1}(x_j)$, for all $j \in \{1, \ldots, n\}$; the result of this one-step rewrite is the term $[\{\!\{s_{i+1}, v_{i+1}(x_1), \ldots, v_{i+1}(x_n)\}\!\}]_A$.

By mathematical induction, it follows that there always exists a minimal PTA-computation with the properties listed in the lemma, that corresponds to the PTA run $\rho \in \mathit{runs}(A)$. Furthermore, in a PTA-step $\widetilde{\varphi}_i$ of the form (12) all terms are completely determined by the PTA run ρ, including the probabilistic substitutions $[\rho]_A$ in the label β_i. It follows that the minimal PTA-computation corresponding to ρ is also unique and the notation $\widetilde{\rho}$ is now well-defined. □

Lemma 5 allows us to define a (total) function $\xi : \mathit{runs}(A) \to \mathit{PTA}\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A))$ which takes each run $\rho \in \mathit{runs}(A)$ and maps it to its associated minimal PTA-computation $\xi(\rho) = \widetilde{\rho} \in \mathit{PTA}\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A))$. Conversely, the following lemma says that, to each minimal PTA-computation of $\Psi_{PTA}(A)$, there corresponds a unique run of the PTA A. This proves that there also exists a functional relation from the set $\mathit{PTA}\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A))$ to the set $\mathit{runs}(A)$.

Lemma 3. *Let A be a probabilistic timed automaton. To each minimal PTA-computation $\widetilde{\rho} = \{\widetilde{\varphi}_i\}_{i \geq 0}$ in the set $\mathit{PTA}\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A))$, with the term $\widetilde{\varphi}_i$ of the form (10), there corresponds a unique PTA run $\rho \in \mathit{runs}(A)$ of the form (8), such that for all $i \geq 0$:*

- *the state s_i coincides with the term σ_i;*
- *the clock valuation v_i satisfies $v_i(x_j) = y_i^j$ for all $j \in \{1, \ldots, n\}$;*
- *the time instant t_{i+1} is given by $\sum_{k=0}^{i} r_k$, i.e., the PTA A and the PRTRT $\Psi_{PTA}(A)$ take the instantaneous, probabilistic transitions at the same time;*
- *the probability distribution p_{i+1} is the probability distribution in $\mathrm{prob}(\sigma_i)$ denoted by the label π_i of a probabilistic rewrite rule r of the form (5), with r contained in the label β_i of the corresponding E/A-canonical one-step instantaneous rewrite.*

Proof. Let $\widetilde{\rho}$ be an arbitrary PTA-computation in $\Psi_{PTA}(A)$ and denote by $\beta_i = ([\mathbb{C}]_A, r, [\theta]_A, [\rho]_A)$ the label of the i-th E/A-canonical one-step instantaneous rewrite in $\widetilde{\rho}$, where the probabilistic rewrite rule r is labeled by the probability distribution π_i. From the assumptions in this lemma it follows that the PTA run corresponding to $\widetilde{\rho}$, whose existence and uniqueness we need to prove, must be of the form

$$(\sigma_0, 0) \xrightarrow[r_0]{\pi_1} (\sigma_1, v_1) \xrightarrow[r_0+r_1]{\pi_2} (\sigma_2, v_2) \xrightarrow[r_0+r_1+r_2]{\pi_3} \ldots \tag{13}$$

such that the clock valuation $v_i : \mathcal{X} \to [0, \infty)$ is given by $v_i(x_j) = y_i^j$, for all $i \geq 1$ and all $j \in \{1, \ldots, n\}$. We use mathematical induction on the number of steps of the PTA run (13), to prove that it can be obtained in A. For the base case, consider the first step $(\sigma_0, 0) \xrightarrow[r_0]{\pi_1} (\sigma_1, v_1)$ of the PTA run, corresponding to the first PTA-step of the minimal PTA-computation $\widetilde{\rho}$:

$$\widetilde{\varphi}_0 : \ [\{\!\{\sigma_0, 0, \ldots, 0\}\!\}]_A \xrightarrow[r_0]{\alpha_0} [\{\!\{\sigma_0, r_0, \ldots, r_0\}\!\}]_A \xrightarrow[\phi(0)]{\beta_0} \left[\{\!\{\sigma_1, y_1^1, \ldots, y_1^n\}\!\}\right]_A . \tag{14}$$

By looking at the timed and probabilistic transitions of the first step of the PTA run separately, we notice that:

i) The clocks in $\mathcal{X}$ satisfy the invariant condition $\mathrm{inv}(\sigma_0)$, before taking the instantaneous transition to σ_1, since this is exactly the condition in the rule of the form (7), corresponding to the tick application in (14), namely the condition that $v_0 + r_0 = (r_0, \ldots, r_0) \in \mathrm{inv}(\sigma_0)$;

ii) The probability distribution $\pi_1 \in \mathrm{prob}(\sigma_0)$ is enabled at time r_0 in A, i.e., it satisfies the enabling condition $v_0 + r_0 = (r_0, \ldots, r_0) \in \tau_{\sigma_0}(\pi_1)$, since this is exactly the condition of the instantaneous, probabilistic rewrite rule r in the label β_0 of (14). Furthermore, in the PTA-step (14), the transition to the timed state $[\{\!\{\sigma_1, y_1^1, \ldots, y_1^n\}\!\}]_A$ is selected with nonzero probability $\Gamma_{\sigma_0}(\pi_1)(\sigma \mapsto \{\!\{\sigma_1, y_1^1, \ldots, y_1^n\}\!\})$, on account of equation (6). It follows that there exists a probabilistic choice in A which selects the instantaneous transition to σ_1 and resets the clocks in $X = \{x_k \mid y_1^k = 0,\ 1 \leq k \leq n\} \subseteq \mathcal{X}$, with the same probability $\pi_1(\sigma_1, X) > 0$, such that $v_1(x_j) = y_1^j$ for all $j \in \{1, \ldots, n\}$.

The rewrite rule conditions, which we refer to in points i) and ii) above, are satisfied since $\widetilde{\rho}$ is an "admissible" minimal PTA-computation in $\Psi_{PTA}(A)$, i.e., it satisfies the formal semantics of PRTRTs. More precisely, the matches made along $\widetilde{\rho}$ satisfy the definition of generalized R/A-matches in the formal semantics of PRTRTs (Definition 5, assumption 3 that $E \cup A \vdash \theta(cond)$), which ensures that both the invariant and the enabling conditions are satisfied when A makes a probabilistic-timed transition. Therefore, the first step of the PTA run (13) can indeed be obtained in A.

Now let $i \geq 1$ be arbitrary and assume, by the induction hypothesis, that the i steps prefix

$$(\sigma_0, 0) \xrightarrow[r_0]{\pi_1} (\sigma_1, v_1) \xrightarrow[r_0+r_1]{\pi_2} \dots \xrightarrow[\sum_{k=0}^{i-1} r_k]{\pi_i} (\sigma_i, v_i)$$

of the PTA run (13) can be obtained in A. We must then show that the next step of the run,

$$(\sigma_i, v_i) \xrightarrow[\sum_{k=0}^{i} r_k]{\pi_{i+1}} (\sigma_{i+1}, v_{i+1}),$$

corresponding to the PTA-step

$$\widetilde{\varphi}_i : \ \left[\{\sigma_i, y_i^1, \dots, y_i^n\}\right]_A \xrightarrow[r_i]{\alpha_i} \left[\{\sigma_i, y_i^1 + r_i, \dots, y_i^n + r_i\}\right]_A \xrightarrow[\phi(0)]{\beta_i} \left[\{\sigma_{i+1}, y_{i+1}^1, \dots, y_{i+1}^n\}\right]_A, \tag{15}$$

can also be obtained in A. Similar to the base case $i = 1$, and due to the fact that $\widetilde{\rho}$ is an "admissible" minimal PTA-computation whose PTA-steps satisfy the PRTRT semantics (Definition 5, assumption 3), we notice that:

i) The clocks in $\mathcal{X}$ satisfy the invariant condition $\text{inv}(\sigma_i)$, before taking the instantaneous transition to σ_{i+1}, since this is the same as the condition in the rule of the form (7), corresponding to the last tick application in (15), namely that $v_i + r_i = v_i + (t_{i+1} - t_i) \in \text{inv}(\sigma_i)$.

ii) The probability distribution $\pi_{i+1} \in \text{prob}(\sigma_i)$ is enabled at time $\sum_{k=0}^{i} r_k$ in A, i.e., it satisfies the enabling condition $v_i + r_i = v_i + (t_{i+1} - t_i) \in \tau_{\sigma_i}(\pi_{i+1})$, since this is the same as the condition of the instantaneous, probabilistic rewrite rule r in the label β_i of (15). Moreover, in the PTA-step (15), the transition to the timed state $[\{\sigma_{i+1}, y_{i+1}^1, \dots, y_{i+1}^n\}]_A$ is selected with nonzero probability $\Gamma_{\sigma_i}(\pi_{i+1})(\sigma \mapsto \{\sigma_{i+1}, y_{i+1}^1, \dots, y_{i+1}^n\})$, as given by equation (6). Therefore, there exists a probabilistic choice in A which selects the instantaneous transition to σ_{i+1} and resets the clocks in $X = \{x_k \mid y_{i+1}^k = 0,\ 1 \leq k \leq n\} \subseteq \mathcal{X}$, with the same probability $\pi_{i+1}(\sigma_{i+1}, X) > 0$, such that $v_{i+1}(x_j) = y_{i+1}^j$ for all $j \in \{1, \dots, n\}$.

It follows that the $(i+1)$-th step of the PTA run (13) can also be obtained in A. By mathematical induction, we have therefore shown that the whole PTA run (13) can indeed be obtained in A. Furthermore, in a PTA run of the form (13) all terms are completely determined by the PTA computation $\widetilde{\rho}$, including the probability distributions π_i in the label of each probabilistic-timed transition of (13). Since we proved the existence of PTA runs of the form (13), corresponding to PTA-computations $\widetilde{\rho}$ with PTA-steps of the form (10), it follows that such PTA runs are also unique, and the notation ρ (removing the tilde from $\widetilde{\rho}$) is therefore well-defined. □

Lemma 3 allows us to define a (total) function $\eta : \mathit{PTA}\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A)) \to \mathit{runs}(A)$ that maps each minimal PTA-computation $\widetilde{\rho} \in \mathit{PTA}\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A))$ to its associated PTA run $\eta(\widetilde{\rho}) = \rho \in \mathit{runs}(A)$. The following is the main result of this section and concludes our correctness proof for the translation of PTA into PRTRTs. It states that to each run ρ of the PTA A there corresponds a unique (minimal) PTA-computation of $\Psi_{PTA}(A)$, up to the indistinguishability relation over PTA-computations, and vice-versa.

Theorem 1. *The set $\mathit{runs}(A)$ of all runs of a PTA A is in one-to-one correspondence with the set $\mathit{PTA}\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A))$ of minimal PTA-computations of its corresponding PRTRT representation $\Psi_{PTA}(A)$.*

Proof. This follows directly from Lemma 5, Lemma 3 and the fact that for any run ρ of the PTA A and any minimal PTA-computation $\widetilde{\rho}$ of $\Psi_{PTA}(A)$, we have $\xi(\rho) = \widetilde{\rho}$ if and only if $\eta(\widetilde{\rho}) = \rho$, which implies that η is the inverse of the function ξ. It follows that both ξ and η are invertible, and therefore bijective. Hence the sets $\mathit{runs}(A)$ and $\mathit{PTA}\text{–}\mathcal{C}_{min}(\Psi_{PTA}(A))$ are in one-to-one correspondence. □

6.2 Stochastic Automata

In *stochastic automata* (SA) [12, 11], the time between transitions is random, following an *arbitrary* probability distribution. This is in contrast with Markov models, where the probability distribution of time transitions can only belong to the exponential family. More precisely, an SA is an automaton where the transitions have the form $s \xrightarrow{a,X} s'$, with a an action and X a set of *timers*. A transition is enabled when all the timers in X have expired, and time cannot advance when there is an enabled transition. An SA may also have nondeterministic behaviors, since multiple transitions may become enabled at the same time, so that any of them can be applied. As the result of taking the transition $s \xrightarrow{a,X} s'$, the automaton goes to state s' and each timer $x \in \kappa(s')$ is assigned a value sampled from its cumulative distribution function $F(x)$.

Formally, let $\mathcal{P}_{\text{fin}}(X)$ denote the set of all *finite* subsets of a set X.

Definition 22. *An SA [12] is a tuple* $(S, s_0, \mathcal{X}, \text{Act}, \longrightarrow, \kappa, F)$ *where:*

- S *is a set of* states *and* $s_0 \in S$ *is the* initial state*;*
- $\mathcal{X}$ *is a set of* timers*;*
- Act *is a set of* actions*;*
- $\longrightarrow \subseteq S \times (\text{Act} \times \mathcal{P}_{\text{fin}}(\mathcal{X})) \times S$ *is the set of* transitions *of the automaton, where we write* $s \xrightarrow{a,X} s'$ *whenever* $(s, a, X, s') \in \longrightarrow$ *and call* $X \subseteq \mathcal{X}$ *the* trigger set *of the transition;*
- $\kappa : S \to \mathcal{P}_{\text{fin}}(\mathcal{X})$ *is the* timer setting function*;*
- $F : \mathcal{X} \to CDF(\mathbb{R})$ *assigns a cumulative distribution function* $F(x)$ *to each timer* $x \in \mathcal{X}$*, with the property that* $F(x)(t) = 0$ *for all* $t < 0$*. For all states* $s \in S$*, the initial value of each timer* $x \in \kappa(s)$ *is a random variable distributed according to* $F(x)$*.*

Example 4. In Fig. 2 we consider an SA modeling a stack that is operated randomly through `push` and `pop` operations. In this simple example, we only consider two timers x, y that start counting down from values sampled from two different CDFs $F(x)$ and $F(y)$, where $F(x)(t) = 1 - \exp\left(-\frac{1}{12}\ t\right)$ is an exponential distribution with a mean of 12 time units and $F(y)(t) = \begin{cases} 0, & t < 7 \\ 1, & t \geq 7 \end{cases}$ is a distribution which assigns a value of 7 time units with probability 1 to timer y, i.e., $F(y)$ acts like a deterministic timer reset. Each state is depicted as a rectangle, with the symbol of the state inside it, as well as the set of timers that are to be reset in that state. There is also an arrow without a source going into the initial state. The SA starts in state s_0 and, since $\kappa(s_0) = \{x, y\}$, it uses the distribution functions $F(x)$ and $F(y)$ to assign starting values to timers x and y, respectively. The automaton makes a transition to state s_1 as soon as the timer x expires, which is 12 time units on average. The expiration of x triggers the `push` action upon the stack. In the new state s_1, the timer x is assigned a new value sampled from the exponential distribution function $F(x)$, while y is not assigned any value and continues to count down. At this point, there are two possible transitions, either to s_2 with a `pop` action, or to s_3 with a `push` action. Notice that the triggering sets of both transitions are the same, which causes a *nondeterministic choice* between them as soon as the timer y expires. Another interesting feature is that, provided that the automaton reaches state s_2 from s_3, the timer x is expired, due to the triggering set $\{x, y\}$ associated with the transition. This causes the next transition from s_2 back to s_0 to be *instantaneous*. The behavior of the automaton in the other states is similar to the one described for states s_0 and s_1.

Translation into PRTRT. We outline the PRTRT representation $\Psi_{SA}(A) = (\Sigma, \emptyset, E, L, R, \pi, \phi, \tau)$ of a finitary stochastic automaton $A = (S, s_0, \mathcal{X}, \text{Act}, \longrightarrow, \kappa, F)$, where $\mathcal{X}$ is a finite set $\{x_1, \ldots, x_n\}$. As in the case of the PRTRT representation of PTA, the MEL theory (Σ, E) contains an equational axiomatization of a computable subfield of the reals that defines the sort `Time`, denoting the time domain. The signature Σ also contains a sort `SAState` with a constant s for each state $s \in S$, a constant `init` of sort `GlobalSystem` and a $(|\mathcal{X}| + 1)$-ary operator `_ , _ , ... , _ : SAState Time ... Time -> System`. The "timed state" of A is represented in $\Psi_{SA}(A)$ by a term $\{s, r_1, r_2, \ldots, r_n\}$, where s is a constant denoting the current state of the SA, and r_i is the current value of the timer x_i. The following probabilistic rewrite rule randomly selects the initial timer values

$$\texttt{rl [init]: init => } \{s_0\texttt{ , }y_1\texttt{ , ... , }y_n\texttt{\}}\ \texttt{with probability}\ y_{k_1} := F_{k_1}\ \texttt{and ... and}\ y_{k_m} := F_{k_m}\ . \tag{16}$$

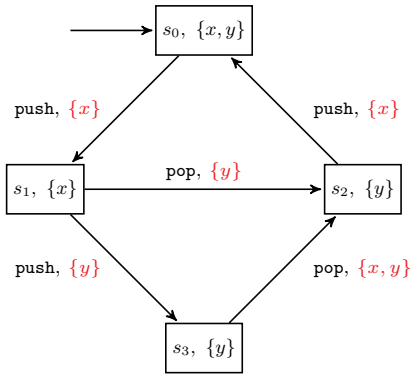

Fig. 2. Simple stochastic automaton with two timers, modeling a stack with random `push` / `pop` operations

where F_i mimics the CDF $F(x_i)$ of timer x_i (see Section 3), $k_1, \ldots, k_m \in \{1, \ldots, n\}$ are some indices such that $\kappa(s_0) = \{x_{k_1}, \ldots, x_{k_m}\}$ and for all $i \in \{1, \ldots, n\} \setminus \{k_1, \ldots, k_m\}$ we have $y_i = 0$. Each transition $s \xrightarrow{a,X} s'$ of the SA is translated into a labeled probabilistic rewrite rule

$$\texttt{rl [}a\texttt{]: \{}s, r_1, \ldots, r_n\texttt{\} => \{}s', r'_1, \ldots, r'_n\texttt{\} with probability } r'_{j_1} := F_{j_1} \texttt{ and ... and } r'_{j_l} := F_{j_l} \texttt{ .} \tag{17}$$

where r_i is 0 if $x_i \in X$ and is a variable y_i otherwise, $\kappa(s') = \{x_{j_1}, x_{j_2}, \ldots, x_{j_l}\}$, and for all indices $i \in \{1, \ldots, n\}$ such that $x_i \notin \kappa(s')$, we have $r'_i = r_i$. Time elapse is modeled by the following tick rule which can advance time until the next timer expires, but only if no transition is enabled in the current state σ

$$\begin{aligned}&\texttt{crl [tick]: \{}\sigma, y_1, \ldots, y_n\texttt{\} => \{}\sigma, \texttt{max}(0, y_1 - y), \ldots, \texttt{max}(0, y_n - y)\texttt{\} in time } y \\ &\qquad \texttt{if } y \texttt{ <= nextTimerExpires(}y_1, \ldots, y_n\texttt{) and not transEnabled(}\sigma, y_1, \ldots, y_n\texttt{) .}\end{aligned} \tag{18}$$

where σ, y_i, and y are all variables, `nextTimerExpires(`$y_1, \ldots, y_n$`)` returns the smallest non-zero value of the y_i, or $+\infty$ if $y_1 = \ldots = y_n = 0$, and `transEnabled(`$\sigma, y_1, \ldots, y_n$`)` holds if and only if some transition is enabled in the given state. The latter function is defined by an equation

$$\texttt{eq transEnabled(}s, r_1, \ldots, r_n\texttt{) = true .}$$

where r_i is 0 if $x_i \in X$ and is a variable y_i otherwise, for each transition $s \xrightarrow{a,X} s'$, and by having an equation that states that *otherwise* (i.e., if none of the above equations apply), `transEnabled(`$\sigma, y_1, \ldots, y_n$`)` is `false`

$$\texttt{eq transEnabled(}\sigma, y_1, \ldots, y_n\texttt{) = false [owise] .}$$

where σ and $y_1, \ldots, y_n$ are variables. '`owise`' is a Maude construct, but it is explained in [10] that it is not an extra-logical feature; any Maude specification with the `owise` construct is equivalent to a theory without this construct.

Example 5. The SA in Fig. 2 is therefore represented as a PRTRT as follows

```
rl [init]: init => {s0, y1, y2} with probability y1 := F1 and y2 := F2 .
rl [push]: {s0, 0, y2} => {s1, y1', y2} with probability y1' := F1 .
rl [pop] : {s1, y1, 0} => {s2, y1, y2'} with probability y2' := F2 .
rl [push]: {s1, y1, 0} => {s3, y1, y2'} with probability y2' := F2 .
rl [push]: {s2, 0, y2} => {s0, y1', y2'} with probability y1' := F1 and y2' := F2 .
rl [pop] : {s3, 0, 0} => {s2, 0, y2'} with probability y2' := F2 .

crl [tick]: {σ, y1, y2} => {σ, max(0, y1 - y), max(0, y2 - y)} in time y
            if y <= nextTimerExpires(y1, y2) and not transEnabled(σ, y1, y2) .
```

where y_1, y'_1, y_2, y'_2, y and σ are variables, while `transEnabled` and `nextTimerExpires` are defined by:

```
eq transEnabled(s0, 0, y2) = true .  eq transEnabled(s1, y1, 0) = true .
eq transEnabled(s2, 0, y2) = true .  eq transEnabled(s3, 0, 0) = true .
eq transEnabled(σ, y1, y2) = false [owise] .

eq nextTimerExpires(y1, y2) = if y1 == 0 then (if y2 == 0 then INF else y2 fi)
                                         else (if y2 == 0 then y1 else min(y1, y2) fi) .
```

Correspondence Theorem. Similar to the case of probabilistic timed automata in Section 6.1, we represent the current "timed state" of a stochastic automaton $A = (S, s_0, \mathcal{X}, \text{Act}, \longrightarrow, \kappa, F)$ as a pair $(s, v) \in S \times \mathbb{R}^{\mathcal{X}}$, with $s \in S$ a state and $v : \mathcal{X} \to [0, \infty)$ a timer valuation.

Definition 23. *A* run *of a SA A is an* infinite *labeled sequence of timed states, of the form*

$$\rho : \ (s_0, v_0) \xrightarrow[t_1]{a_1, X_1} (s_1, v_1) \xrightarrow[t_2]{a_2, X_2} (s_2, v_2) \xrightarrow[t_3]{a_3, X_3} \dots \tag{19}$$

where $v_0(x)$ is sampled from the CDF F_x for each timer $x \in \mathcal{X}$, and for all $i \geq 1$:

- *$s_{i-1} \xrightarrow{a_i, X_i} s_i$ is a transition of A;*
- *$t_i \in \mathbb{R}_{\geq 0}$ gives the current time, right before the SA makes an instantaneous, probabilistic transition to state s_i, with the convention that $t_0 = 0$;*
- *all timers $x \in X_i$ must satisfy $v_{i-1}(x) = t_i - t_{i-1}$, i.e., all timers in X_i must expire at time t_i, right before making the transition to s_i;*
- *for all timers $x \in \kappa(s_i)$, i.e., x is a timer to be reset in s_i, the value of $v_i(x)$ is a sample from the CDF F_x, otherwise v_i must satisfy $v_i(x) = \max\{0,\, v_{i-1}(x) - (t_i - t_{i-1})\}$.*

We write runs(A) for the set of all runs of a stochastic automaton A.

We now define elementary computation steps in the PRTRT representation $\Psi_{SA}(A)$, corresponding to single steps in a run of A.

Definition 24. *An* SA-step *φ in $\Psi_{SA}(A)$ is given by a finite (possible empty) sequence of E/A-canonical one-step* tick *rewrites, followed by an E/A-canonical one-step* instantaneous *rewrite.*

Based on our translation of SA into PRTRTs given in the previous paragraph, any SA-step must have either one of the two following forms:

1. A single application of the instantaneous, probabilistic rewrite rule (16), denoted

$$\varphi_{-1} : \ \texttt{init} \xrightarrow[\phi(0)]{\beta_{-1}} \left[\!\left\{\sigma_0, y_0^1, \dots, y_0^n\right\}\!\right]_A, \tag{20}$$

   with no applications of the tick rule (18), where σ_0 is a term of sort `SAState`, $y_0^1, \dots, y_0^n$ are terms of sort `Time`, and β_{-1} is the label of the corresponding E/A-canonical one-step instantaneous rewrite. An SA-step of the form (20) sets the values of the timers in the initial state $\left[\!\left\{\sigma_0, y_0^1, \dots, y_0^n\right\}\!\right]_A$.

2. A finite (possibly empty) sequence of applications of the tick rule (18), followed by a single application of the instantaneous, probabilistic rewrite rule (17),

$$\begin{aligned}\varphi_i : \ \left[\!\left\{\sigma_i, y_i^1, \dots, y_i^n\right\}\!\right]_A &\xrightarrow[r_i^1]{\beta_i^1} \left[\!\left\{\sigma_i, \texttt{max}(0, y_i^1 - r_i^1), \dots, \texttt{max}(0, y_i^n - r_i^1)\right\}\!\right]_A \xrightarrow[r_i^2]{\beta_i^2} \\ &\xrightarrow[r_i^2]{\beta_i^2} \left[\!\left\{\sigma_i, \texttt{max}(0, y_i^1 - (r_i^1 + r_i^2)), \dots, \texttt{max}(0, y_i^n - (r_i^1 + r_i^2))\right\}\!\right]_A \xrightarrow[r_i^3]{\beta_i^3} \dots \\ &\xrightarrow[r_i^{k_i}]{\beta_i^{k_i}} \left[\!\left\{\sigma_i, \texttt{max}\left(0, y_i^1 - \left(\textstyle\sum_{j=1}^{k_i} r_i^j\right)\right), \dots, \texttt{max}\left(0, y_i^n - \left(\textstyle\sum_{j=1}^{k_i} r_i^j\right)\right)\right\}\!\right]_A \xrightarrow[\phi(0)]{\beta_i} \\ &\xrightarrow[\phi(0)]{\beta_i} \left[\!\left\{\sigma_{i+1}, y_{i+1}^1, \dots, y_{i+1}^n\right\}\!\right]_A, \end{aligned} \tag{21}$$

   where σ_i, σ_{i+1} are terms of sort `SAState`, $y_i^1, \dots, y_i^n$ and $y_{i+1}^1, \dots, y_{i+1}^n$ are terms of sort `Time`, $k_i \geq 0$ is the number of tick rewrites in φ_i with $r_i^1, \dots, r_i^{k_i}$ their associated durations and $\beta_i^1, \dots, \beta_i^{k_i}$ the corresponding labels of the E/A-canonical one-step tick rewrites, and β_i is the label of the last E/A-canonical one-step instantaneous rewrite in φ_i, which contains an instantaneous probabilistic rewrite rule of the form (17) labeled by `[`a`]`, associated with an SA transition $\sigma_i \xrightarrow{a, X} \sigma_{i+1}$.

Notice that, in order to arrive at the form (21), we make use of the relation

$$\max\{0, \max\{0, a\} - b\} = \max\{0, a - b\}$$

for all real values a, b with $b \geq 0$. For example, since $r_i^1, r_i^2 \geq 0$ it follows that the state

$$\left[\{\sigma_i, \mathtt{max}(0,\, \mathtt{max}(0, y_i^1 - r_i^1) - r_i^2), \ldots, \mathtt{max}(0,\, \mathtt{max}(0, y_i^n - r_i^1) - r_i^2)\}\right]_A$$

reached from the state $\left[\{\sigma_i, y_i^1, \ldots, y_i^n\}\right]_A$ after two applications of the tick rewrite rule (18) labeled by β_i^1 and β_i^2, with corresponding durations r_i^1 and r_i^2, is the same as the state reached after a single application of (18) with duration $r_i^1 + r_i^2$:

$$\left[\{\sigma_i, \mathtt{max}(0, y_i^1 - (r_i^1 + r_i^2)), \ldots, \mathtt{max}(0, y_i^n - (r_i^1 + r_i^2))\}\right]_A .$$

It can be shown by mathematical induction over the length of φ_i that it can indeed be written as (21).

Definition 25. *A* minimal *SA-step is an SA-step which is either of the form* (20)*, or of the form* (21) *with* $k_i = 1$ *and* $r_i^1 \neq \phi(0)$*, or with* $k_i = 0$*.*

We now introduce an equivalence relation over SA-steps which allows us to only focus on minimal SA-steps when reasoning about arbitrary SA-steps in $\Psi_{SA}(A)$, since they are *indistinguishable* from other non-minimal, equivalent SA-steps.

Definition 26. *The SA-steps*

$$\varphi:\ [u_0]_A \xrightarrow[r_1]{\beta_1} [u_1]_A \xrightarrow[r_2]{\beta_2} \ldots \xrightarrow[r_n]{\beta_n} [u_n]_A \xrightarrow[\phi(0)]{\beta} [v]_A$$

and

$$\varphi':\ [u'_0]_A \xrightarrow[r'_1]{\beta'_1} [u'_1]_A \xrightarrow[r'_2]{\beta'_2} \ldots \xrightarrow[r'_m]{\beta'_m} [u'_m]_A \xrightarrow[\phi(0)]{\beta'} [v']_A$$

are said to be indistinguishable*, and we write* $\varphi \sim \varphi'$ *if their associated finite (possibly empty) timed computation paths* $[u_0]_A \xrightarrow[r_1]{} \ldots \xrightarrow[r_n]{} [u_n]_A$ *and* $[u'_0]_A \xrightarrow[r'_1]{} \ldots \xrightarrow[r'_m]{} [u'_m]_A$ *are indistinguishable (see Definition 9), and* $\beta = \beta'$*, which also implies* $[v]_A = [v']_A$*.*

It follows easily that the indistinguishability relation $\sim$ over the set of all SA-steps in $\Psi_{SA}(A)$ is an equivalence relation.

Lemma 4. *Let* φ *be an SA-step in* $\Psi_{SA}(A)$*. Then there exists a unique minimal SA-step* φ^{min} *which is indistinguishable from* φ*.*

Proof. Let φ be an arbitrary SA-step in $\Psi_{SA}(A)$. If φ is of the form (20) then it is already minimal, and we can pick $\varphi^{min} = \varphi$. If φ is of the form (21), we use the fact that any application of the tick rewrite rule (18) in φ_i only changes the timer terms $y_i^1, \ldots, y_i^n$, without changing the current state term σ_i. Furthermore, since φ_i is an "admissible" SA-step, which can be obtained in $\Psi_{SA}(A)$, it follows that the conditions of all conditional tick rewrite rules in φ_i are satisfied. Therefore, we can safely replace the series of timed transitions in φ_i with a single one, whose duration is given by the sum of durations in the original series. If all timed transitions in φ_i have zero duration, we can entirely remove the series of timed transitions, so that φ_i reduces to a single, instantaneous transition. Hence, we have two separate cases:

1. If $\sum_{j=1}^{k_i} r_i^j \neq \phi(0)$, we associate with $\varphi = \varphi_i$ the minimal SA-step

$$\varphi^{min} = \varphi_i^{min}:\ \left[\{\sigma_i, y_i^1, \ldots, y_i^n\}\right]_A \xrightarrow[r_i]{\alpha_i} \left[\{\sigma_i, \mathtt{max}(0, y_i^1 - r_i), \ldots, \mathtt{max}(0, y_i^n - r_i)\}\right]_A \xrightarrow[\phi(0)]{\beta_i} \left[\{\sigma_{i+1}, y_{i+1}^1, \ldots, y_{i+1}^n\}\right]_A \quad (22)$$

   which is indistinguishable from φ_i, where the label α_i of the E/A-canonical one-step tick rewrite contains the substitution $\theta = \left\{y_1 \mapsto y_i^1,\ \ldots,\ y_n \mapsto y_i^n,\ y \mapsto \sum_{j=1}^{k_i} r_i^j\right\}$ for matching against the tick rewrite rule (18), and $r_i = \sum_{j=1}^{k_i} r_i^j$ is the total duration of the E/A-canonical one-step tick rewrites in φ_i.

2. If $\sum_{j=1}^{k_i} r_i^j = \phi(0)$, the minimal SA-step corresponding to $\varphi = \varphi_i$ is given by:

$$\varphi^{min} = \varphi_i^{min} : \ \left[\{\sigma_i, y_i^1, \ldots, y_i^n\}\right]_A \xrightarrow[\phi(0)]{\beta_i} \left[\{\sigma_{i+1}, y_{i+1}^1, \ldots, y_{i+1}^n\}\right]_A . \tag{23}$$

In all cases, since φ_i^{min} is completely determined by the terms and labels of φ_i, it follows that $\varphi_i^{min} = \varphi^{min}$ is also unique. □

Notice that the mapping $\varphi \mapsto \varphi^{min}$ is surjective, but it is not injective since there exist at least two distinct SA-steps, with the same total (nonzero) duration, which are mapped into the same corresponding minimal SA-step. We next define SA-computations as infinite sequences of SA-steps in $\Psi_{SA}(A)$, preceded by an application of the rule (16) to initialize all the timers.

Definition 27. *An* SA-computation *of the PRTRT representation $\Psi_{SA}(A)$ of a stochastic automaton A is an infinite sequence $\{\varphi_i\}_{i\geq -1}$ of SA-steps in $\Psi_{SA}(A)$, where φ_{-1} is of the form* (20) *and φ_i is of the form* (21) *for all $i \geq 0$. We denote by SA–$\mathcal{C}(\Psi_{SA}(A))$ the set of all SA-computations of the probabilistic real-time rewrite theory $\Psi_{SA}(A)$.*

Similar to the case of SA-steps, we can define minimal SA-computations as follows.

Definition 28. *A* minimal *SA-computation is an SA-computation $\{\varphi_i\}_{i\geq -1}$ such that all its SA-steps are minimal. We denote by SA–$\mathcal{C}_{min}(\Psi_{SA}(A))$ the set of all minimal SA-computations of the PRTRT $\Psi_{SA}(A)$.*

Since φ_{-1} is always minimal, in order to show that $\{\varphi_i\}_{i\geq -1}$ is a minimal SA-computation, it suffices to check that all SA-steps, starting with φ_0, are minimal. The indistinguishability relation generalizes from SA-steps to SA-computations in a natural way.

Definition 29. *Two SA-computations $\{\varphi_i\}_{i\geq -1}$ and $\{\varphi'_i\}_{i\geq -1}$ are* indistinguishable *if and only if the SA-step φ_i is indistinguishable from φ'_i for all $i \geq -1$. In this case, we write $\{\varphi_i\}_{i\geq -1} \sim \{\varphi'_i\}_{i\geq -1}$.*

Notice that $\varphi_0 \sim \varphi'_0$ also implies $\varphi_{-1} \sim \varphi'_{-1}$, and therefore $\{\varphi_i\}_{i\geq -1}$ and $\{\varphi'_i\}_{i\geq -1}$ are indistinguishable if and only if $\varphi_i \sim \varphi'_i$ for all $i \geq 0$ (no need to check the case $i = -1$). The relation $\sim$ is an equivalence over the set of all SA-computations SA–$\mathcal{C}(\Psi_{SA}(A))$. As in the case of SA-steps, given an SA-computation $\{\varphi_i\}_{i\geq -1}$, there exists exactly one minimal SA-computation $\{\varphi_i^{min}\}_{i\geq -1}$, also denoted $\{\varphi_i\}_{i\geq -1}^{min}$, which is indistinguishable from $\{\varphi_i\}_{i\geq -1}$. Furthermore, the mapping $\{\varphi_i\}_{i\geq -1} \mapsto \{\varphi_i\}_{i\geq -1}^{min}$ is surjective, but it is not injective since there exist at least two distinct SA-computations whose SA-steps have the same corresponding total (nonzero) duration, and which are mapped into the same minimal SA-computation.

Given two SA-computations $\{\varphi_i\}_{i\geq -1}$ and $\{\varphi'_i\}_{i\geq -1}$, they are indistinguishable if and only if they have the same corresponding minimal SA-computation $\{\varphi_i\}_{i\geq -1}^{min} = \{\varphi'_i\}_{i\geq -1}^{min}$. This follows from the transitivity of the indistinguishability relation and the fact that indistinguishable SA-computations have the same total duration. Therefore, each minimal SA-computation uniquely determines an equivalence class of indistinguishable SA-computations, and the quotient set SA–$\mathcal{C}(\Psi_{SA}(A))/_\sim$ can be indexed by the elements of SA–$\mathcal{C}_{min}(\Psi_{SA}(A))$ as follows:

$$\text{SA–}\mathcal{C}(\Psi_{SA}(A))/_\sim = \{\, [\rho]_\sim \mid \rho \in \text{SA–}\mathcal{C}_{min}(\Psi_{SA}(A)) \,\}. \tag{24}$$

The set SA–$\mathcal{C}_{min}(\Psi_{SA}(A))$ therefore contains the "essential" SA-computations of $\Psi_{SA}(A)$, i.e., all other SA-computations are indistinguishable from the ones in this set. The following lemma claims that, to each run of a stochastic automaton A there corresponds a unique *minimal* SA-computation of $\Psi_{SA}(A)$, which shows that there exists a *functional relation* from the set *runs*(A) of all runs of A, to the set SA–$\mathcal{C}_{min}(\Psi_{SA}(A))$ of all minimal SA-computations of $\Psi_{SA}(A)$.

Lemma 5. *Let A be a stochastic automaton. To each run $\rho \in runs(A)$ of A of the form (19) there corresponds a unique minimal SA-computation $\widetilde{\rho} = \{\widetilde{\varphi}_i\}_{i\geq -1}, \in$ SA–$\mathcal{C}_{min}(\Psi_{SA}(A))$, such that for all $i \geq 0$:*

- *the term σ_i coincides with the state s_i of the SA, i.e., $\sigma_i = s_i$;*
- *the term y_i^j of sort* `Time` *coincides with the value of the timer x_j in the timer valuation v_i of the SA, i.e., $y_i^j = v_i(x_j)$ for all $j \in \{1, \ldots, n\}$;*
- *the duration term r_i is given by the time difference $t_{i+1} - t_i$, i.e., the stochastic automaton A and the PRTRT $\Psi_{SA}(A)$ take the instantaneous, probabilistic transitions at the same time;*

– *the label β_i of the E/A-canonical one-step instantaneous rewrite contains a probabilistic rewrite rule of the form (17) labeled by* $[a_{i+1}]$*, corresponding to the transition* $s_i \xrightarrow[t_{i+1}]{a_{i+1},X_{i+1}} s_{i+1}$ *of* A.

Proof. Let $\rho \in runs(A)$ be an arbitrary run of the stochastic automaton A. For all $i \geq 0$, the SA-step $\widetilde{\varphi}_i$ of the minimal SA-computation $\{\widetilde{\varphi}_i\}_{i\geq 0}$ corresponding to ρ, which satisfies the properties in the lemma, must be of the form

$$\begin{aligned}\widetilde{\varphi}_i :\ & [\{s_i, v_i(x_1), \ldots, v_i(x_n)\}]_A \\ & \xrightarrow[t_{i+1}-t_i]{\alpha_i} [\{s_i,\, \mathtt{max}(0, v_i(x_1) - (t_{i+1} - t_i)),\, \ldots,\, \mathtt{max}(0, v_i(x_n) - (t_{i+1} - t_i))\}]_A \\ & \xrightarrow[\phi(0)]{\beta_i} [\{s_{i+1}, v_{i+1}(x_1), \ldots, v_{i+1}(x_n)\}]_A\,, \end{aligned} \quad (25)$$

with the instantaneous probabilistic rewrite rule in β_i of the form (17) and labeled by $[a_{i+1}]$. Also, the SA-step $\widetilde{\varphi}_{-1}$ must be of the form

$$\widetilde{\varphi}_{-1} :\ \mathtt{init} \xrightarrow[\phi(0)]{\beta_{-1}} [\{s_0, v_0(x_1), \ldots, v_0(x_n)\}]_A\,, \quad (26)$$

with the label β_{-1} containing an instantaneous probabilistic rewrite rule of the form (16). We prove that such a minimal SA-computation can indeed be obtained in $\Psi_{SA}(A)$ by using generalized R/A-matches and applying E/A-canonical one-step rewrites. We use mathematical induction on the number of SA-steps of $\{\widetilde{\varphi}_i\}_{i\geq -1}$, similarly to the proof of Lemma 5. □

6.3 Deterministic and Stochastic Petri Nets

Deterministic and stochastic Petri nets (DSPNs) [2] are a fairly general class of timed Petri nets, where a transition can fire after having been continuously enabled for either a fixed (deterministic) or a random (exponentially distributed) amount of time. DSPNs are strictly more expressive than generalized stochastic Petri nets [3] (and, hence, stochastic Petri nets), which can be seen as DSPNs in which all deterministic transitions are instantaneous.

There exist many variations of the basic model, including having inhibitor arcs, arc multiplicities that are functions of the marking, transition precedences, etc. To focus on the real-time and probabilistic aspects of the model, we assume a "standard" Petri net model extended with the above firing delays, and refer to [33] for the treatment of advanced Petri net features in rewriting logic.

Definition 30. *A DSPN [2] is a tuple (P, T, F, τ, R) where:*

– *P is a finite set of* places*;*
– *$T = T^D \uplus T^S$ is a finite set of* transitions*, partitioned into sets T^D and T^S of* deterministic *and* stochastic *transitions, respectively, and satisfying $T \cap P = \emptyset$;*
– *$F \subseteq (P \times T) \cup (T \times P)$ is the* flow relation*;*
– *$\tau : T^D \to \mathbb{R}_{\geq 0}$ is a function that maps each* deterministic *transition t to its firing delay $\tau(t)$; if $\tau(t) = 0$ we call t a* instantaneous *transition;*
– *$R : T^S \to \mathbb{R}$ is a function that associates to each* stochastic *transition t the rate parameter $R(t)$ of the exponential distribution followed by the firing delay of t.*

A transition must fire when it has been enabled continuously for the duration of its firing delay. We assume that the "enabled-time" of a transition is reset to zero when the transition is applied, even though the transition could have been enabled with multiple disjoint submarkings.

Example 6. We consider specifying a **M/D/2/5/5** queueing model of a client-server architecture, as a DSPN. We use an extended form of Kendall's notation [17] to represent the following:

– the amount of time until a client sends a service request to the server is Markovian, i.e., exponentially distributed (**M**),
– the time of service is deterministic (**D**),
– there are exactly two servers that can resolve clients' requests (**2**),
– there is a maximum number of five customers allowed in the system at any time, either waiting to be served or in service (**5**),
– there are exactly five clients that may send a request to the server (**5**).

We have provided a representation of this queueing model in Fig. 3, which also shows the *initial marking* of the Petri net, with five clients in place p_1, that are neither requesting nor receiving service, and two servers in place p_4, ready to answer requests from the clients. Each place is represented by a circle, which may contain none or several tokens inside it, along with the evolution of the Petri net. The stochastic transitions are shown as empty rectangles (t_1), the deterministic ones are depicted as filled rectangles (t_3), while the instantaneous transitions are shown as thick lines (t_2). Apart from showing a label below each transition, we provide further information above them, i.e., for the stochastic transitions we show the rate of the exponential distribution associated with the waiting times of the tokens inside *any* of the corresponding precondition sets, while for the deterministic transitions we display the fixed amount of time associated with them.

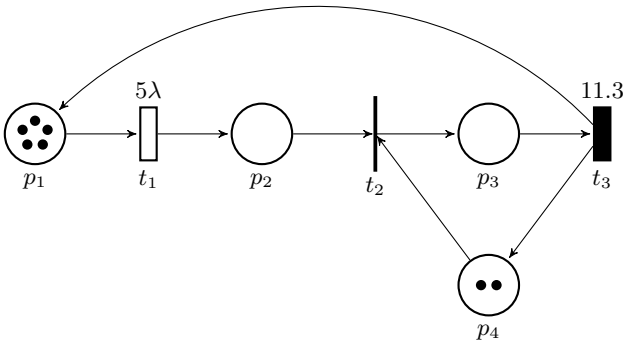

Fig. 3. A **M/D/2/5/5** queueing model of a client-server architecture with 5 clients and 2 servers, specified as a deterministic and stochastic Petri net.

Thus, in Fig. 3, the waiting time until *either one* of the five clients in place p_1 issues a service request is exponentially distributed with rate parameter 5λ, where $\lambda \in (0, \infty)$ is the rate of the exponential distribution followed by the waiting time of *a single* client. As soon as a client issues a request, its associated token moves to p_2, where it may be joined by a token in p_4. The tokens in p_4 represent idle servers that are ready to resolve the client's request. The two tokens may then be removed from their places p_2, p_4 and joined into a single token in p_3, after the instantaneous transition t_2. The tokens in p_3 represent clients during the service time. The transition t_3 takes 11.3 time units, which represents the total service time of the client undergoing the transition. After transition t_3 is made, a token goes back to p_4 to denote the fact that the associated server is again ready to handle client requests, while the token corresponding to the client moves to p_1, representing a message from the server to the client, acknowledging that the request has been resolved. Note that the DSPN model is *concurrent*, i.e., either one of the clients may send requests to the servers or may be served, if a server is available, independent of what the other clients are doing.

The formal model corresponding to this example is given by the DSPN (P, T, F, τ, R) where:

- $P = \{p_1, p_2, p_3, p_4\}$,
- $T = \{t_1, t_2, t_3\} = T^D \uplus T^S$ such that $T^D = \{t_2, t_3\}$ and $T^S = \{t_1\}$,
- the flow relation is $F = \{(p_1, t_1), (t_1, p_2), (p_2, t_2), (p_4, t_2), (t_2, p_3), (p_3, t_3), (t_3, p_1), (t_3, p_4)\}$,
- the delays of each transition are specified by $\tau : \{t_2, t_3\} \to \mathbb{R}_{\geq 0}$ with $\tau(t_2) = 0$, $\tau(t_3) = 11.3$, as well as $R : \{t_1\} \to \mathbb{R}$ with $R(t_1) = 5\lambda$.

Translation into PRTRT. We explain now how a DSPN can be mapped into a PRTRT, and we prove the correctness of this mapping in Theorem 2. Our representation follows the approach of [25], which sees a marking as a multiset of places and a transition as a multiset rewrite rule. In addition, for each transition $t \in T$, we associate a timer that denotes the remaining time during which the transition must be continuously enabled to fire. Such a timer can be represented by a term $\texttt{<}\ t\ \texttt{;}\ r\ \texttt{>}$, where t is the transition and r is its timer value. The global state of the system is therefore represented as a term $\texttt{\{}m\texttt{\}}$, where m is a multiset of places and transition-timers, with multiset union denoted by juxtaposition. As a result

of firing a transition, other previously enabled transitions may be disabled, and vice versa. Therefore, we must recompute all the transition timer values when a transition fires.

For each transition t in the DSPN with pre-set $p_1\ p_2\ \ldots\ p_m$ and post-set $q_1\ q_2\ \ldots\ q_n$, we therefore have a rewrite rule

```
rl [applyTrans-t] :
   {< t ; 0 >  p1 p2 ... pm  REST}  =>  {recomputeTimers(< t ; INF >  q1 q2 ... qn  REST)} .
```

which fires the transition t when its timer is 0. As a result, the pre-set is removed from the state, the post-set is added to it, t's timer is turned off (although it may be reset by `recomputeTimers` if the transition is still enabled, or re-enabled), and the function `recomputeTimers` is applied to the entire resulting state to recompute all transition timer values.

The function `recomputeTimers` is defined as follows: (i) if a transition is enabled and the corresponding timer is turned off (i.e., has the value `INF`), then the timer is reinitialized to the firing delay of the transition, otherwise the timer is left unchanged; and (ii) if a transition is not enabled, its timer is turned off. Case (i) can easily be done by an equation defining `recomputeTimers` for *deterministic* transitions. However, an *equation* defining `recomputeTimers` cannot reset the timer of a stochastic transition, since the new timer value should be assigned probabilistically. Therefore, the timer of the stochastic transition is initialized to a new value `reset`, which will be replaced by a probabilistically chosen value in a *rewrite rule*. That is, for any *deterministic* transition t, case (i) above is defined by the following equation:

```
eq recomputeTimers(< t ; TI >  p1 p2 ... pm REST)
 = < t ; if TI == INF then τ(t) else TI fi >  recomputeTimers(p1 p2 ... pm REST) .
```

and for any *stochastic* transition t, case (i) is defined by the following equation:

```
eq recomputeTimers(< t ; TI >  p1 p2 ... pm REST)
 = < t ; if TI == INF then reset else TI fi >  recomputeTimers(p1 p2 ... pm REST) .
```

For each stochastic transition t with rate $R(t)$ we therefore have a rewrite rule

```
rl [set-stoch-timer] : < t ; reset >  =>  < t ; X >  with probability X := ExpRate(R(t)) .
```

where the function `ExpRate(λ)` mimics the CDF of the exponential distribution with rate parameter $\lambda \in \mathbb{R}$. For case (ii), if the transition is not enabled, the following `owise` equation sets the corresponding timer to `INF`:

```
eq recomputeTimers(< T ; TI >  REST)  =  < T ; INF >  recomputeTimers(REST) [owise] .
```

where `T` is a variable. Finally, we add the tick rule

```
crl [tick] : {SYSTEM}  =>  {decreaseTimers(SYSTEM, Y)} in time Y if Y <= mte(SYSTEM) .
```

where `decreaseTimers` decreases the value of each timer by the elapsed time `Y`, and `mte` gives the smallest timer value (or 0 if a timer has the value `reset`). Therefore, this tick rule may advance time until the next transition timer expires. Considering that $T^D = \{t_1^D, \ldots, t_u^D\}$ and $T^S = \{t_1^S, \ldots, t_w^S\}$ for fixed positive integers u, w, the initial marking of the DSPN is given by a constant `initMarking` of sort `GlobalSystem` which is set equationally through

```
eq initMarking = recomputeTimers(p1 ... pm
  < t1D ; INF > ... < tuD ; INF > < t1S ; INF > ... < twS ; INF >)
```

corresponding to an initial marking given by the multiset of places $p_1\ \ldots\ p_m$, together with the multiset of all transition-timer pairs whose timers are initially switched off. Appendix B gives a more detailed specification of this PRTRT representation of a DSPN, including the declarations of all sorts, variables, etc.

Example 7. The representation of the DSPN in Fig. 3 contains the instantaneous rules

```
rl [applyTrans-t1] : {< t1 ; 0 > p1 REST}  =>  {recomputeTimers(< t1 ; INF > p2 REST)} .
rl [applyTrans-t2] : {< t2 ; 0 > p2 p4 REST}  =>  {recomputeTimers(< t2 ; INF > p3 REST)} .
rl [applyTrans-t3] : {< t3 ; 0 > p3 REST}  =>  {recomputeTimers(< t3 ; INF > p1 p4 REST)} .
```

together with the equations defining the `recomputeTimers` function

```
eq recomputeTimers(< t1 ; TI >  p1 REST)
  =  < t1 ; if TI == INF then reset else TI fi >  recomputeTimers(p1 REST) .

eq recomputeTimers(< t2 ; TI >  p2 p4 REST)
  =  < t2 ; if TI == INF then 0 else TI fi >  recomputeTimers(p2 p4 REST) .

eq recomputeTimers(< t3 ; TI >  p3 REST)
  =  < t3 ; if TI == INF then 11.3 else TI fi >  recomputeTimers(p3 REST) .

eq recomputeTimers(< t ; TI >  REST)  =  < t ; INF >  recomputeTimers(REST) [owise] .

ceq recomputeTimers(REST)  =  REST  if noTimers(REST) .
```

as well as the tick rule and the rule for setting the timer of the stochastic transition t_1 to a value sampled from the exponential distribution with rate 5:

```
rl [set-stoch-timer] : < t1 ; reset >  =>  < t1 ; X >  with probability X := ExpRate(5) .
```

Correspondence Theorem. In order to formally define a run of a DPSN, we take inspiration from the timed transition systems in [22], where transitions are also required to stay enabled for a certain amount of time before they can fire; however this amount can only be deterministic, and cannot be randomly selected. Denote by $S^{\oplus}$ the set of all finite multisets over a set S and by $\emptyset_S$ the empty multiset over S.

Definition 31. *A* run *of a DSPN $D = (P, T, F, \tau, R)$ is an infinite labeled sequence of the form*

$$\rho: \quad (m_0, r_0) \longrightarrow (m_1, r_1) \longrightarrow (m_2, r_2) \longrightarrow \dots \tag{27}$$

where $r_0 = 0$, and for all $i \geq 1$, $m_i \in P^{\oplus}$ denotes a marking of D, i.e., a multiset of places, and $r_i \in \mathbb{R}_{\geq 0}$ is the time at which the marking m_i occurs. The time divergence condition $\lim_{j\to\infty} r_j = \infty$ must hold, i.e., all runs are considered to be non-Zeno. Furthermore, there are two kinds of transitions from (m_i, r_i) to (m_{i+1}, r_{i+1}), for all $i \geq 1$:

1. Tick transitions, *when $m_i = m_{i+1} = m$ and $r_i < r_{i+1}$, and we denote them by $(m, r_i) \xrightarrow{tick} (m, r_{i+1})$.*
2. Instantaneous transitions, *when $r_i = r_{i+1} = r$ and the marking m_{i+1} is obtained from m_i as a result of firing some transition $t \in T$, such that the transition t has been uninterruptedly enabled for exactly $\gamma(t)$ time units along the run ρ, where $\gamma : T \to \mathbb{R}_{\geq 0}$ is the function assigning to each transition its actual delay, defined through*

$$\gamma(t) = \begin{cases} \tau(t), & t \in T^D, \\ \overline{X}(t), & t \in T^S, \end{cases}$$

with $\overline{X}(t)$ a sample from the exponential distribution with rate parameter $R(t)$. More precisely, there exists $j \leq i$ such that $r_i - r_j = \gamma(t)$, t is enabled with all markings $m_j, m_{j+1}, \dots, m_i$, and only instantaneous transitions are allowed from m_i. We denote instantaneous transitions of D by $(m_i, r) \xrightarrow{t} (m_{i+1}, r)$,

Let runs(D) be the set of all runs of a DSPN D.

We now define elementary computation steps in $\Psi_{DSPN}(D)$, which correspond to single steps in a run of a DSPN D.

Definition 32. *A* DSPN-step *in the PRTRT representation $\Psi_{DSPN}(D)$ of D is given by a finite sequence of zero or more applications of the* `set-stochastic-timer` *rule, followed by a finite sequence of E/A-canonical one-step* tick *rewrites, i.e., applications of the given tick rule, and ending with a E/A-canonical one-step* instantaneous *rewrite, corresponding to an application of the* `applyTrans-t` *rule. A DSPN-step therefore has the form $[u]_A \xrightarrow{*} [u']_A \xrightarrow[\tau]{*} [u'']_A \xrightarrow{\beta} [v]_A$, which we also denote by $[u]_A * \xrightarrow[\tau]{\beta} [v]_A$, where $[u]_A$, $[u']_A$, $[u'']_A$ and $[v]_A$ are fully simplified terms of sort* `GlobalSystem`.

A *DSPN-computation* in $\Psi_{DSPN}(D)$, simulating a run ρ of D, is an infinite sequence $\widetilde{\rho}$ of DSPN-steps in $\Psi_{DSPN}(D)$ and we denote by $DSPN\text{–}\mathcal{C}(\Psi_{DSPN}(D))$ the set of all DSPN-computations of the probabilistic real-time rewrite theory $\Psi_{DSPN}(D)$. The following result shows that DSPNs are faithfully represented in our PRTRT formalism, by providing a bijection between the sets of runs of each model.

Theorem 2. *Let D be a deterministic and stochastic Petri net. There exists a bijection between the sets runs(D) and DSPN–$\mathcal{C}(\Psi_{DSPN}(D))$, associating to each run of D a unique DSPN-computation.*

Proof. (Sketch) Firstly, notice that the initial term in the run sequence of D is $(p_1 \dots p_m, 0)$ if and only if the initial term `initMarking`, from which a DSPN-computation of D starts, is given by

$$\texttt{\{recomputeTimers(}p_1 \ \dots \ p_m \ \texttt{<}\ t_1^D \ \texttt{;}\ \texttt{INF >}\ \dots \ \texttt{<}\ t_u^D \ \texttt{;}\ \texttt{INF >}\ \texttt{<}\ t_1^S \ \texttt{;}\ \texttt{INF >}\ \dots \ \texttt{<}\ t_w^S \ \texttt{;}\ \texttt{INF >)\}}$$

and this is true, since it is set equationally in our translation. We now treat the cases of tick and instantaneous transitions separately. A run of D contains a tick transition of the form

$$(p_1 \dots p_m, r) \xrightarrow{\text{tick}} (p_1 \dots p_m, r + \Delta r)$$

if and only if the DSPN-computation of $\Psi_{DSPN}(D)$ contains a finite sequence of zero or more applications of the `set-stoch-timer` rule, rewriting the term

$$[u]_A = \texttt{\{}p_1 \ \dots \ p_m \ \texttt{<}\ t_1 \ \texttt{;}\ v_1 \ \texttt{>}\ \dots \ \texttt{<}\ t_j \ \texttt{;}\ v_j \ \texttt{>}\ \texttt{<}\ t_{j+1} \ \texttt{;}\ \texttt{reset >}\ \dots \ \texttt{<}\ t_{j+k} \ \texttt{;}\ \texttt{reset >\}}$$

to a term

$$[u']_A = \texttt{\{}p_1 \ \dots \ p_m \ \texttt{<}\ t_1 \ \texttt{;}\ v_1 \ \texttt{>}\ \dots \ \texttt{<}\ t_j \ \texttt{;}\ v_j \ \texttt{>}\ \texttt{<}\ t_{j+1} \ \texttt{;}\ \overline{X}(t_{j+1}) \ \texttt{>}\ \dots \ \texttt{<}\ t_{j+k} \ \texttt{;}\ \overline{X}(t_{j+k}) \ \texttt{>\}}$$

where $t_{j+1}, \dots, t_{j+k}$ are all stochastic transitions and $\overline{X}(t_{j+h}) = v_{j+h}$ is, as above, a sample from the exponential distribution with rate $R(t_{j+h})$, followed by a finite sequence of applications of the tick rule in our translation, obtaining the term:

$$[u'']_A = \texttt{\{}p_1 \ \dots \ p_m \ \texttt{<}\ t_1 \ \texttt{;}\ v_1 - \Delta r \ \texttt{>}\ \dots \ \texttt{<}\ t_j \ \texttt{;}\ v_j - \Delta r \ \texttt{>}\ \texttt{<}\ t_{j+1} \ \texttt{;}\ v_{j+1} - \Delta r \ \texttt{>}\ \dots \ \texttt{<}\ t_{j+k} \ \texttt{;}\ v_{j+k} - \Delta r \ \texttt{>\}}$$

Also, a run of D contains an instantaneous transition of the form $(p_1 \ \dots \ p_m, r) \xrightarrow{t} (q_1 \ \dots \ q_n, r)$ if and only if there exists a E/A-canonical one-step instantaneous rewrite in the DSPN-computation of $\Psi_{DSPN}(D)$, of the form $[u'']_A \xrightarrow{\beta} [v]_A$, rewriting the term

$$[u'']_A = [\texttt{\{<}\ t \ \texttt{;}\ 0 \ \texttt{>}\ \ p_1 \ \dots \ p_m \ \ \texttt{REST\}}]_A$$

to the term

$$[v]_A = [\texttt{\{recomputeTimers(<}\ t \ \texttt{;}\ \texttt{INF >}\ \ q_1 \ \dots \ q_n \ \ \texttt{REST)\}}]_A$$

with $\beta = ([\mathbb{C}]_A, r, [\theta]_A, [\rho]_A)$ the label of the E/A-canonical one-step instantaneous rewrite, where $\mathbb{C} = \odot$, r is the instantaneous rewrite rule `applyTrans-`t, θ is a substitution such that the multiset union `REST` θ(`REST`) represents the entire marking together with the set of all transition-timer pairs, and defines the current state in the DSPN-computation; also $\rho = \emptyset$, since no probabilistic substitutions are involved in this transition.

Finally, the equations defining the `recomputeTimers` operation ensure that transitions stay enabled for an amount of time equal to their corresponding delay, and therefore this property, which the run of D satisfies, is also preserved in the PRTRT representation $\Psi_{DSPN}(D)$ of D. □

6.4 Handling Uncertainty in Probabilistic Transitions

We have identified two models for probabilistic real-time systems that allow the probability distribution associated with a transition to be nondeterministically chosen from a *set* of probability distributions that satisfy some constraints. However, these are not higher-order constraints since probability distributions can in general be parameterized and the constraints may then refer to these parameters and become simple equality or inequality constraints. Therefore, we can also represent these models as PRTRTs, which implies that PRTRTs are strictly more expressive than (untimed) probabilistic rewrite theories in which the probability distribution is deterministically chosen.

We first recall the definition of Markov decision processes (MDP), first introduced in [6], which are a building block of both formalisms that we describe in the following sections.

Definition 33. *An MDP [6] is a tuple $\mathcal{M} = (S, \text{Act}, A, p)$ where:*

- *S is a finite set of states;*
- *Act is a set of actions;*
- *$A : S \to \mathcal{P}(\text{Act})$ is a function giving the set of actions $A(s) \subseteq \text{Act}$ available at s, for each $s \in S$;*
- *$p = \{p_{s,a} : S \to [0,1] \mid s \in S,\ a \in A(s)\}$ is a family of probability mass functions such that $p_{s,a}(s') \in [0,1]$ is the probability of making a transition from state s to state s', under an action a which is available at s.*

A *discrete-time Markov chain* is an MDP with $\text{Act} = \{a\}$ containing a single action, and with $A(s) = \{a\}$ for all states $s \in S$, i.e., action a is available in all states.

Timed Probabilistic Transition Systems. In the *timed probabilistic transition systems* (TPTS) of [35] the probability of making a transition belongs to an *interval*. This can be modeled in our formalism by exploiting the fact that, for a given rewrite rule r, π_r is a family of probability distributions, indexed both by the substitutions for the variables in the lefthand side of r and by the substitutions for the nondeterministically instantiated variables in the righthand side of r.

Denote by $\mathcal{I}$ the set of all closed subintervals of the unit interval, except for the singleton $\{0\}$, i.e., $\mathcal{I} = \{[a,b] \mid 0 \leq a \leq b \leq 1,\ b \neq 0\}$.

Definition 34. *A TPTS [35] is a tuple* $(S, \mathrm{Act}, s_0, \longrightarrow, \delta)$ *where:*

- S *is a set of states with* $s_0 \in S$ *the initial state;*
- Act *is a set of actions;*
- $\longrightarrow \subseteq S \times \mathrm{Act} \times S$ *is a labeled transition relation giving the nonprobabilistic transitions;*
- $\delta : S \times S \to \mathcal{I}$ *is a partial, unlabeled transition function that defines the probabilistic timed transitions—the probability of going from some state* s *to another state* s' *in* one unit of time *lies within the interval* $\delta(s, s')$.

It is assumed that if $\delta(s, s')$ *is defined for some states* s, s'*, then no labeled transition* $s \xrightarrow{a} s'$ *exists; i.e., transitions between any two states are either probabilistic (unlabeled) or nonprobabilistic (labeled).*

Following [35], given a state $s \in S$ we define the set $\Delta(s)$ containing all the probability mass functions $f_s : S \to [0,1]$ with the property that $f_s(s') \in \delta(s, s')$ for all states $s' \in S$ such that $\delta(s, s')$ is defined, and with:

$$\sum_{\substack{s' \in S \\ \delta(s,s') \text{ is defined}}} f_s(s') = 1.$$

In other words, $\Delta(s)$ is the set of all valid probability mass functions over the set of successor states from s, which satisfy the interval constraints defined by the δ function. By selecting some $f_s \in \Delta(s)$ for each state $s \in S$, we obtain a possible *instance* of the TPTS, which can be seen as a Markov decision process; if there are no labeled transitions, this instance becomes a discrete-time Markov chain.

Translation into PRTRT. In what follows, we provide the PRTRT representation $\Psi_{TPTS}(T)$ of a TPTS $T = (S, \mathrm{Act}, s_0, \longrightarrow, \delta)$. In our PRTRT encoding we add the nondeterministically selected probability mass functions to the state. The current state of the TPTS is therefore represented with the syntax `{`s`;`f`}` where $s \in S$ is a state and $f \in \Delta(s'')$ is a probability mass, with s'' a predecessor state of s, i.e., the TPTS reached state s from s'' by sampling from f. The initial state is set through the equation:

`eq initState = {`s_0`;` $\emptyset$`} .`

Let φ be a variable over the set of probability mass functions on S. To each labeled transition $s \xrightarrow{a} s'$ we associate an instantaneous, nonprobabilistic labeled rewrite rule:

`rl [`a`] : {`s`;` φ`} => {`s'`;` φ`} .`

We also add a probabilistic tick rewrite rule to our representation

`crl [tick`$_S$`] :`
`{`s`;` φ`} => {`σ`;` f_s`} in time 1 if` $f_s \in \Delta(s)$ `with probability` σ `:=` f_s `.`

for each state s with the property that $\delta(s, s')$ is defined for at least one state s'. The special feature of this tick rule, compared to the ones in the PRTRT representation of the formalisms in the previous sections, is that the probability mass function f_s is nondeterministically chosen from the set $\Delta(s)$, for each state $s \in S$. By this we mean that the probability values defined by f_s are nondeterministically chosen to satisfy the given interval constraints, and therefore the set membership $f_s \in \Delta(s)$ is not a higher-order construct. More precisely, the condition $f_s \in \Delta(s)$ in the above tick rule is translated into a condition only referring to the values $f_s(s')$ for all $s' \in S$, which can easily be expressed in membership equational logic:

$$f_s \in \Delta(s) \quad \Leftrightarrow \quad \left[\bigwedge_{\delta(s,s') \text{ is defined}} f_s(s') \in \delta(s, s') \right] \wedge \left[\sum_{\substack{s' \in S \\ \delta(s,s') \text{ is defined}}} f_s(s') = 1 \right].$$

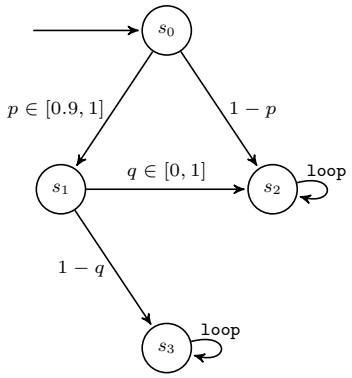

Fig. 4. A simple timed probabilistic transition system with a family $\mathcal{B}(p)$ of Bernoulli distributions over the states $\{s_1, s_2\}$ when coming from state s_0, and another family of Bernoulli distributions $\mathcal{B}(q)$ over $\{s_2, s_3\}$ when going from state s_1.

Example 8. We consider a simple TPTS with set of states $S = \{s_0, s_1, s_2, s_3\}$, Act $= \{\texttt{loop}\}$, looping labeled transitions $s_2 \xrightarrow{\texttt{loop}} s_2$, $s_3 \xrightarrow{\texttt{loop}} s_3$, and with the δ function defined by $\delta(s_0, s_1) = [0.9, 1]$, $\delta(s_0, s_2) = [0, 0.1]$ and $\delta(s_1, s_2) = \delta(s_1, s_3) = [0, 1]$; this TPTS is also depicted in Fig. 4. The labeled transitions are encoded in the PRTRT representation as the following two labeled rewrite rules:

```
rl [loop] : {s2 ; r}  =>  {s2 ; r} .
rl [loop] : {s3 ; r}  =>  {s3 ; r} .
```

where r is a variable. The set $\Delta(s_0)$ contains all probability mass functions $f_{s_0} : S \to [0, 1]$ such that $f_{s_0}(s_1) \in [0.9, 1]$, $f_{s_0}(s_2) \in [0, 0.1]$ and $f_{s_0}(s_1) + f_{s_0}(s_2) = 1$, i.e., f_{s_0} is an arbitrary Bernoulli distribution $\mathcal{B}(p) = \begin{pmatrix} s_1 & s_2 \\ p & 1-p \end{pmatrix}$ with parameter $p = f_{s_0}(s_1) \in [0.9, 1]$. Therefore, the set $\Delta(s_0)$ can be identified with the family of Bernoulli distributions over $\{s_1, s_2\}$ with the parameter p ranging over $[0.9, 1]$, i.e., $\Delta(s_0) = \{\mathcal{B}(p) \mid p \in [0.9, 1]\}$. Similarly, $\Delta(s_1)$ is identified with the family of Bernoulli distributions $\mathcal{B}(q)$ over $\{s_2, s_3\}$, indexed by the parameter $q \in [0, 1]$. Finally, we add the following two probabilistic tick rewrite rules to our PRTRT representation

```
crl [tick s0] :
    {s0 ; r}  =>  {σ ; p}  in time 1
    if p ∈ [0.9, 1]  with probability σ := (s1  s2 ; p  1 - p) .

crl [tick s1] :
    {s1 ; r}  =>  {σ ; q}  in time 1
    if q ∈ [0, 1]  with probability σ := (s2  s3 ; q  1 - q) .
```

where σ, p, q, r are variables. Notice how the set membership conditions $f_{s_0} \in \Delta(s_0)$ and $f_{s_1} \in \Delta(s_1)$ are translated into the simple interval constraints $p \in [0.9, 1]$ and $q \in [0, 1]$, on the parameters p and q of the corresponding Bernoulli distributions.

Correspondence Theorem. We now prove that any TPTS T can be mapped into a corresponding PRTRT $\Psi_{TPTS}(T)$, by providing an explicit bijection between the sets of runs of each model. The current *timed state* of T is given by a pair $(s, t) \in S \times \mathbb{N}$.

Definition 35. *A* run *of a TPTS T is any infinite labeled sequence of nondeterministic and probabilistic transitions between timed states, of the form $(s, t) \xrightarrow{a} (s', t)$ and $(s, t) \xrightarrow{f_s} (s', t+1)$, respectively, such that $s \xrightarrow{a} s'$ is a labeled transition of T and $f_s \in \Delta(s)$ is a probability mass function with $f_s(s') > 0$. It is also assumed that the first timed state of any run is $(s_0, 0)$. We write runs(T) for the set of runs of a TPTS T.*

Example 9. A possible run of the TPTS in Fig. 4, instantiating the parameters of the Bernoulli distributions, is the following:

$$\rho : \ (s_0, 0) \xrightarrow{\begin{pmatrix} s_1 & s_2 \\ 0.95 & 0.05 \end{pmatrix}} (s_1, 1) \xrightarrow{\begin{pmatrix} s_2 & s_3 \\ 0.7 & 0.3 \end{pmatrix}} (s_3, 2) \xrightarrow{\texttt{loop}} (s_3, 2) \xrightarrow{\texttt{loop}} \ \ldots$$

We now define elementary computation steps in $\Psi_{TPTS}(T)$, corresponding to single steps in a run of a TPTS T.

Definition 36. *A* TPTS-step *in the PRTRT representation $\Psi_{TPTS}(T)$ is either an E/A-canonical one-step* instantaneous *rewrite of the form $\{s\,;\,\varphi\} \xrightarrow{\alpha} \{s'\,;\,\varphi\}$ obtained by applying the rule r labeled by a, corresponding to the labeled transition $s \xrightarrow{a} s'$ of T, or an E/A-canonical one-step* tick *rewrite of the form $\{s\,;\,\varphi\} \xrightarrow[1]{\beta} \{s'\,;\,f_s\}$, always with a duration of one time unit, obtained by applying the probabilistic tick rule corresponding to a timed probabilistic transition of T, such that $\delta(s,s')$ is defined.*

A *TPTS-computation* in $\Psi_{TPTS}(T)$, simulating a run of T, is an infinite sequence of TPTS-steps in $\Psi_{TPTS}(T)$. Denote by *TPTS–*$\mathcal{C}(\Psi_{TPTS}(T))$ the set of all TPTS-computations of the probabilistic real-time rewrite theory $\Psi_{TPTS}(T)$.

Theorem 3. *Let T be a timed probabilistic transition system. There exists a bijection between the sets runs(T) and TPTS–*$\mathcal{C}(\Psi_{TPTS}(T))$*, associating to each run of T a unique TPTS-computation in $\Psi_{TPTS}(T)$.*

Proof. (Sketch) First of all, notice that the initial timed state of T is $(s_0, 0)$ if and only if the initial state $\Psi_{TPTS}(T)$ is $\{s_0\,;\,\emptyset\}$, which is true since this is set equationally. In what follows, let $s, s' \in S$ be arbitrary states and let $t \in \mathbb{N}$ be an arbitrary positive integer. A run of T contains a transition of the form $(s,t) \xrightarrow{a} (s',t)$ if and only if there exists a TPTS-step in $\Psi_{TPTS}(T)$ of the form $\{s\,;\,\varphi\} \xrightarrow{\alpha} \{s'\,;\,\varphi\}$ with $\alpha = ([\mathbb{C}]_A, r, \emptyset, \emptyset)$, where $\mathbb{C} = \odot$ and r is a rewrite rule of the form:

```
rl [a] : {s ; φ}  =>  {s' ; φ} .
```

Also, a run of T contains a probabilistic timed transition of the form $(s,t) \xrightarrow{f_s^*} (s', t+1)$, for some $f_s^* \in \Delta(s)$ with $f_s^*(s') > 0$, if and only if there exists a TPTS-step in $\Psi_{TPTS}(T)$ of the form $\{s\,;\,\varphi\} \xrightarrow[1]{\beta} \{s'\,;\,f_s^*\}$, with $\beta = ([\mathbb{C}]_A, r, [\theta]_A, [\rho]_A)$, where:

- the context $\mathbb{C}$ is simply the hole $\odot$;
- r is a probabilistic tick rewrite rule of the form

```
crl [tick_S] :
    {s ; φ}  =>  {σ ; f_s}  in time 1  if f_s ∈ Δ(s)  with probability σ := f_s .
```

- the substitution $[\theta]_A$ is given by the singleton $\{f_s \mapsto f_s^*\}$;
- the substitution $[\rho]_A$ is the singleton $\{\sigma \mapsto s'\}$, selected with probability $f_s(s')$. □

Timed Probabilistic Systems. In [13] a particular kind of *timed probabilistic systems* (TPS) are introduced in order to define the formal semantics of the more high-level, probabilistic model of *stochastic transition systems*. TPSs are probabilistic timed models in which the time that a node waits in a location—after selecting an outgoing action, but before *performing* the action—is a random value, whose *average* is given, but whose probability distribution is not specified. In this section, we provide an explicit mapping from TPSs into PRTRTs and discuss some computability issues of this mapping.

Let $V = \{v_1, \ldots, v_n\}$ be a set of variables over a set T.

Definition 37. *A* timed probabilistic system *(TPS) [13] is a tuple $(\mathcal{M}, time, I)$ representing a Markov decision process $\mathcal{M} = (S, \mathrm{Act}, A, p)$ together with:*

- *a function $time : S \times \mathrm{Act} \to \mathbb{R}_{\geq 0}$ that associates to each state-action pair (s,a) the average value $time(s,a)$ of the amount of time spent at s before performing action a;*
- *a valuation function $I : S \times V \to T$ that associates to each state $s \in S$ and each variable $v \in V$ the value $I(s,v)$ of v in s.*

A *behavior* of an MDP (S, Act, A, p) is an infinite sequence $\omega :\ s_0\ a_0\ s_1\ a_1\ \ldots$ with $s_i \in S$, $a_i \in A(s_i)$ and $p_{s_i,a_i}(s_{i+1}) > 0$ for all $i \geq 0$. Given a behavior prefix $s_0\ a_0\ \ldots\ s_n$ of the underlying Markov decision process of the TPS, the average time elapsed along this prefix is given by $\sum_{k=0}^{n-1} time(s_k, a_k)$. The amount of time elapsed before action a is performed is hence a random variable $\varphi(s,a) : \Omega \to \mathbb{R}_{\geq 0}$ defined over some probability space $(\Omega, \mathcal{F}, \mathbb{P})$, whose *average value* is given by the function *time*; formally $\mathbb{E}[\varphi(s,a)] = time(s,a)$ for all state-action pairs (s,a), where $\mathbb{E}$ is the so-called expectation operator. The values that $\varphi(s,a)$ assumes are contained in some *given* subset of $\mathbb{R}_{\geq 0}$. However, the exact probability distribution of $\varphi(s,a)$ is unknown, and this represents the main challenge when mapping TPSs into PRTRTs.

Discussion. In the general case, the average value of $\varphi(s,a)$ is defined as the Lebesgue integral $\mathbb{E}[\varphi(s,a)] = \int_{\omega\in\Omega} \varphi(s,a)(\omega)\ \mathbb{P}(\mathrm{d}\omega)$. We therefore denote by $Sol(s,a)$ the set of all probability measures $\mathbb{P} : \mathcal{F} \to [0,1]$ that are solutions to the equation

$$time(s,a) = \int_{\omega\in\Omega} \varphi(s,a)(\omega)\ \mathbb{P}(\mathrm{d}\omega), \tag{28}$$

where $\mathcal{F}$ is a fixed σ-algebra over some given nonempty set Ω. Each such solution $\mathbb{P}$ defines a probability distribution $\mathbb{P} \circ \varphi(s,a)^{-1} : \mathbb{R}_{\geq 0} \to [0,1]$ for the random variable $\varphi(s,a)$, i.e., the probability that $\varphi(s,a) \in E$, for some set $E \subseteq \mathbb{R}_{\geq 0}$ of positive real numbers, is given by:

$$\mathbb{P} \circ \varphi(s,a)^{-1}(E) = \mathbb{P}\left(\{\omega \in \Omega \mid \varphi(s,a)(\omega) \in E\}\right).$$

The set $Sol(s,a)$ is typically uncountably infinite, and therefore not computable. The algebraic real numbers also do not suffice to represent all solutions in $Sol(s,a)$. This is the main reason why TPSs cannot in general be represented by *finite* PRTRT specifications. If we weaken this condition and allow the corresponding PRTRT specification, namely its underlying signature Σ, to be uncountably infinite, then we are able to obtain a representation of TPSs in our formalism. However, such a representation is not computable and therefore cannot be executed or simulated, rendering our PRTRT encoding useful for theoretical purposes only.

Translation into PRTRT. In order to obtain a PRTRT representation of a TPS, a special MEL signature (K, Σ, S) is required. Namely, for each state s and each action a, we consider a sort denoted $Dist(s,a)$, whose corresponding kind we denote by $Dist_k(s,a)$. Furthermore, we consider the set $\Sigma_{\lambda, Dist(s,a)} \in \Sigma$ of constant function symbols of sort $Dist(s,a)$ which correspond to all solutions $\mathbb{P} \in Sol(s,a)$ to equation (28). This set is typically uncountable, rendering the entire MEL signature of the PRTRT uncountable.

The PRTRT representation of TPSs is similar to the one of stochastic automata, in the sense that timers are also used in this case, and the "timed state" of the TPS has the form $\{s, a, s', r, \mathbb{P}\}$, where s is the current state, a is the next action to perform, s' is the probabilistically selected next state, r is a timer giving the remaining time until the automaton performs the action a, and $\mathbb{P} \in Sol(s,a)$ is the nondeterministically chosen probability measure defining the probability distribution $\mathbb{P} \circ \varphi(s,a)^{-1}$ from which the initial value of r was sampled. When the timer r reaches 0, the TPS performs action a and makes a transition from state s to s'. In this new state, the TPS again has to select an action b, set the initial value of the timer in s' according to a probability distribution $\mathbb{P}' \circ \varphi(s',b)^{-1}$, where $\mathbb{P}' \in Sol(s',b)$ is selected nondeterministically, and sample a successor state σ from the probability distribution $p_{s',b} : S \to [0,1]$. Each such series of transitions of a TPS is modeled by a labeled probabilistic rewrite rule of the form

```
crl [a_b]: {s, a, s', 0, P}  =>  {s', b, σ, r, P'}  if P' ∈ Sol(s', b)
          with probability σ := p_{s',b} and r := P' ∘ φ(s', b)^-1  .
```

with s, s', σ, r variables of the appropriate sorts, and with $\mathbb{P}$ of sort $Dist(s,a)$ and $\mathbb{P}'$ of sort $Dist(s',b)$. This rule sets the initial timer value r in state s' to a random value sampled from a nondeterministically chosen probability distribution $\mathbb{P}' \circ \varphi(s',b)^{-1}$.

Similar to stochastic automata, time elapse is modeled by a tick rule of the following form, which advances time by a nondeterministically chosen amount until the timer in the current state expires, but only if no transition from the current state is enabled

```
crl [tick] :
    {s, a, s', r, P}  =>  {s, a, s', max(r − y, 0), P} in time y
    if y <= r and not transEnabled(s,r) .
```

where s, s', a, r, y and $\mathbb{P}$ are variables and `transEnabled(`s`,`r`)` is defined by the equations:

```
eq transEnabled(s,0) = true .
eq transEnabled(s,r) = false [owise] .
```

Finally, the valuation function I is easily specified using a set of equations, i.e., for each state $s \in S$ and each variable $v \in V$, if $I(s,v) = t$ then we add to the PRTRT the equation:

```
eq I(s, v) = t .
```

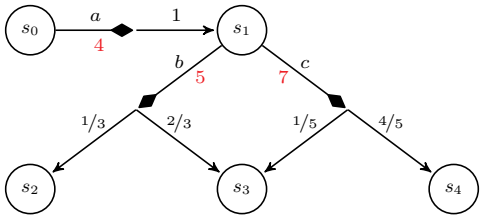

Fig. 5. A simple timed probabilistic system in which the actions are depicted as diamond-tipped arrows, the average time for performing an action is shown in red, and the probabilistic transitions are the arrows with probability values attached to them.

Example 10. We provide an example of a simple TPS, depicted in Fig. 5, with no variables, i.e., $V = \emptyset$, set of states $S = \{s_0, s_1, s_2, s_3, s_4\}$, set of actions $\text{Act} = \{a, b, c\}$, and with $p = \{p_{s_0,a},\ p_{s_1,b},\ p_{s_1,c}\}$ defined through:

$$p_{s_0,a} = \begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \quad p_{s_1,b} = \begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ 0 & 1/3 & 2/3 & 0 \end{pmatrix}, \quad p_{s_1,c} = \begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ 0 & 0 & 1/5 & 4/5 \end{pmatrix}.$$

Also, the average amount of time spent in s_0 for performing action a is $time(s_0, a) = 4$, whereas $time(s_1, b) = 5$ and $time(s_1, c) = 7$. For executability purposes, we may consider that the random variables $\varphi(s, a)$ only assume a finite set of values, for all states $s \in S$ and all actions $a \in \text{Act}$. In particular, we let $\varphi(s_1, b)$ assume a finite set of n given values $\varphi_1, \varphi_2, \ldots, \varphi_n \in \mathbb{R}_{\geq 0}$ with probabilities $p_1, p_2, \ldots, p_n \in [0, 1]$ respectively, where n is a positive integer. In this case, $\varphi(s_1, b)$ is a *discrete* random variable, the Lebesgue integral (28) becomes a finite sum $\sum_{i=1}^{n} \varphi_i p_i$ and each probability distribution for $\varphi(s_1, b)$ is completely defined by the probabilities $p_1, p_2, \ldots, p_n$ that satisfy the linear equation $\sum_{i=1}^{n} \varphi_i p_i = 5$. The conditional instantaneous probabilistic rewrite rule that performs action a when the timer expires in state s_0, and selects b as the next action is then

$$\texttt{crl}\ [a_b]\texttt{:}\ \{s_0, a, s_1, 0, q_1; \ldots; q_m\}\ \texttt{=>}\ \{s_1, b, \sigma, r, p_1; \ldots; p_n\}$$
$$\texttt{if}\ p_1 + \ldots + p_n = 1\ \texttt{and}\ p_1\varphi_1 + \ldots + p_n\varphi_n = 5$$
$$\texttt{with probability}\ \sigma := \begin{pmatrix} s_2 & s_3 \\ 1/3 & 2/3 \end{pmatrix}\ \texttt{and}\ r := \begin{pmatrix} \varphi_1 & \ldots & \varphi_n \\ p_1 & \ldots & p_n \end{pmatrix}.$$

where σ, r, $q_1, \ldots, q_m$ and $p_1, \ldots, p_n$ are variables.

Correspondence Theorem. We now show that any TPS $T = (\mathcal{M}, time, I)$ can be mapped into a PRTRT denoted $\Psi_{TPS}(T)$, by providing a bijection between the sets of runs of each model.

Let $\{\varphi(s, a) : \Omega \to \mathbb{R}_{\geq 0}\}$ be a set of random variables for each state s and each action a available at s, such that the set of values that each variable may assume is given.

Definition 38. *A* run ρ *of a TPS* T *is an infinite labeled sequence of states, of the form*

$$\rho:\ s_1 \xrightarrow[\mathbb{P}_1]{a_1,\ t_1} s_2 \xrightarrow[\mathbb{P}_2]{a_2,\ t_2} s_3 \xrightarrow[\mathbb{P}_3]{a_3,\ t_3} \ldots \tag{29}$$

where, for all $i \geq 0$*:*

- $s_i \in S$ *is a state of* T*;*
- $a_i \in A(s_i)$ *is an action available at* s_i*;*
- $p_{s_i,a_i}(s_{i+1}) > 0$*, i.e., the TPS* T *makes a transition to* s_{i+1} *under action* a_i *with non-zero probability;*
- $\mathbb{P}_i \in Sol(s_i, a_i)$ *is a probability measure which is a solution to (28) with* $s = s_i$ *and* $a = a_i$*;*
- t_i *represents the initial value of the timer at state* s_i*, or the actual amount of time that* T *spends at* s_i *before performing action* a_i*; the value of* t_i *is also a realization of the random variable* $\varphi(s_i, a_i)$ *following the probability distribution* $\mathbb{P}_i \circ \varphi(s_i, a_i)^{-1}$*, and whose average value is given by* $time(s_i, a_i)$*.*

We write $runs(T)$ *for the set of runs of a timed probabilistic system* T*.*

Example 11. Consider the example in Fig. 5, and let the random variable $\varphi(s_1, b)$ be defined over a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with $\Omega = \{\omega_1, \omega_2\}$, and whose possible values are $\varphi(s_1, b)(\omega_1) = 1$, $\varphi(s_1, b)(\omega_2) = 9$, representing the two possible amounts of time that the TPS spends at s_1 before performing action b. A possible run of the TPS is then $\rho:\ s_1 \xrightarrow[\mathbb{P}]{b,\ 9} s_2$, with the probability measure $\mathbb{P} : \mathcal{F} \to [0, 1]$ defined by

$\mathbb{P}(\omega_1) = \mathbb{P}(\omega_2) = 1/2$. The probabilities that $\varphi(s_1, b)$ is either 1 or 9 are therefore both equal to $1/2$. These values for the probability measure are determined by solving the system of linear equations

$$\begin{cases} p_1 + 9p_2 = 5 \\ p_1 + p_2 = 1, \end{cases}$$

together with the constraint $p_1, p_2 \in [0, 1]$, where $p_i = \mathbb{P}(\omega_i)$, $i = 1, 2$.

Definition 39. *A* TPS-step *in $\Psi_{TPS}(T)$ is given by a finite sequence of E/A-canonical one-step* tick *rewrites, i.e., applications of the tick rewrite rule in our PRTRT representation, followed by an E/A-canonical one-step* instantaneous *rewrite, of the form $[u]_A \xrightarrow[\tau]{*} [u']_A \xrightarrow{\beta} [v]_A$, which we also denote by $[u]_A * \xrightarrow[\tau]{\beta} [v]_A$, where $[u]_A$, $[u']_A$, $[v]_A$ are all fully simplified terms of sort* `GlobalSystem`.

A *TPS-computation* in $\Psi_{TPS}(T)$, simulating a run ρ of T, is an infinite sequence $\widetilde{\rho}$ of TPS-steps in $\Psi_{TPS}(T)$. Denote by $TPS\text{–}\mathcal{C}(\Psi_{TPS}(T))$ the set of all TPS-computations of the probabilistic real-time rewrite theory $\Psi_{TPS}(T)$.

Theorem 4. *Let T be a timed probabilistic system. There exists a bijection between the sets runs(T) and $TPS\text{–}\mathcal{C}(\Psi_{TPS}(T))$, associating to each run of T of the form (29) a unique TPS-computation in $\Psi_{TPS}(T)$, of the form:*

$$\widetilde{\rho}: \ [\{s_1, a_1, s_2, t_1, \mathbb{P}_1\}]_A * \xrightarrow[t_1]{\beta_1} [\{s_2, a_2, s_3, t_2, \mathbb{P}_2\}]_A * \xrightarrow[t_2]{\beta_2} \dots \tag{30}$$

Proof. (Sketch) For all $i \geq 1$, a run of T contains a transition of the form $s_i \xrightarrow[\mathbb{P}_i]{a_i,\ t_i} s_{i+1}$ if and only if there exists a TPS-step in $\Psi_{TPS}(T)$ of the form

$$[\{s_i, a_i, s_{i+1}, t_i, \mathbb{P}_i\}]_A * \xrightarrow[t_i]{\beta_i} [\{s_{i+1}, a_{i+1}, s_{i+2}, t_{i+1}, \mathbb{P}_{i+1}\}]_A ,$$

in which t_i is the total duration of the series of tick applications from the lefthand side to the righthand side of the above transition, and $\beta_i = ([\mathbb{C}]_A, r_i, [\theta]_A, [\rho]_A)$ is the label of the associated E/A-canonical one-step instantaneous rewrite, where:

- the context $\mathbb{C}$ is just the hole $\odot$;
- r_i is the instantaneous probabilistic rewrite rule in our translation, labeled by $a_i a_{i+1}$:

  `crl [`$a_i a_{i+1}$`]: {`s_i`,` a_i`,` s_{i+1}`, 0,` $\mathbb{P}_i$`} => {`s_{i+1}`,` a_{i+1}`,` σ`,` r`,` $\mathbb{P}'$`} if` $\mathbb{P}' \in Sol(s_{i+1}, a_{i+1})$
  `with probability` σ `:=` $p_{s_{i+1}, a_{i+1}}$ `and` r `:=` $\mathbb{P}'$ `.`

  corresponding to a transition of the TPS T from state s_{i+1} under action a_{i+1};
- $[\theta]_A$ instantiates the variable $\mathbb{P}'$, i.e., $\theta = \{\mathbb{P}' \mapsto \mathbb{P}_{i+1}\}$, where $\mathbb{P}_{i+1}$ is the probability measure in the run (29) of the TPS T;
- $[\rho]_A$ instantiates both variables σ and r, i.e., $\rho = \{\sigma \mapsto s_{i+2},\ r \mapsto t_{i+1}\}$ for constants s_{i+2} and t_{i+1} of the corresponding sorts; the substitution ρ is selected with probability

  $$\pi_{r_i}([\{\mathbb{P}' \mapsto \mathbb{P}_{i+1}\}]_A)([\{\sigma \mapsto s_{i+2},\ r \mapsto t_{i+1}]_A) = p_{s_{i+1}, a_{i+1}}(s_{i+2}) \cdot \left[\mathbb{P}_{i+1} \circ \varphi(s_{i+1}, a_{i+1})^{-1}(t_{i+1})\right] ,$$

  which is nonzero, from the definition of a TPS run. □

7 Formal Specification of the OGDC Algorithm

This section provides the PRTRT specification of the main part of the Optimal Geographical Density Control (OGDC) algorithm, first introduced in [36]. In previous work, e.g., [30], all probabilistic behaviors were handled in an *ad hoc* way, by means of random sampling and Monte Carlo simulations, which only allowed for partially exploring the state space of the OGDC model. Using PRTRTs we are able to provide the complete formal specification of this algorithm, including the specification of all its probabilistic behaviors.

In what follows we provide a brief introduction to the OGDC algorithm and continue with its PRTRT specification, which is comprised of the same number of 11 rewrite rules as its Real-Time Maude counterpart in [34]. We omit most of the equational specification part, which is the same as in [34] except for two

equations defining the initial values of some backoff timers (T_a and T_b). We also omit the definition of various functions used for geometrical computations and for other purposes, e.g., `findClosestNeighbor`, whose names are self-explanatory.

Given an area to be monitored, also known as a *sensing area*, we consider the scenario in which several wireless sensor nodes are deployed to ensure the coverage and connectivity of this area. *Coverage* refers to the fact that the union of the coverage areas of all active nodes, which can be seen as two-dimensional disks, is a superset of the sensing area. *Connectivity* is ensured as long as all nodes deployed on this area are able to communicate with one another. A *density control process* is an algorithm that selects which nodes to switch on or off in order to maintain these coverage and connectivity properties. The OGDC algorithm is a fully localized, distributed density control process which adds an optimality criterion, namely that the number of active nodes should be kept to a minimum, throughout the lifetime of the network. Geometrically, this condition is equivalent to minimizing the total overlap of the coverage areas of all active nodes. Based on this equivalence, and using standard results of mathematical programming, the authors of [36] obtain explicit conditions for achieving the optimal network configuration. To the best of our knowledge, they were the first to use such explicit conditions in a density control process.

The OGDC algorithm runs in several *rounds*, with each round consisting of a *node selection* phase, followed by a *steady state* phase. In the node selection phase all nodes are initially set to an *undecided* state and the set of working nodes is progressively selected, such that at the end of this phase all nodes are either switched *on* or *off*. This phase begins with all nodes volunteering to be a *starting node* with some prescribed probability. If a node volunteers, then it sets a *backoff timer* whose expiration triggers the node to switch on and to broadcast a power-on message containing its current location and a random direction, uniformly distributed on $[0, 2\pi]$, along which the next working node should be located. If an *undecided* node receives a power-on message, then it first checks whether its entire coverage area is covered by its surrounding active nodes. In this case, the node switches off; otherwise, it sets a backoff timer to a value proportional to the deviation of the node from the *optimal position* w.r.t. the optimality criterion of OGDC, that of minimizing the total overlap of all coverage areas. When the backoff timer of a nonvolunteering node expires, the node broadcasts a power-on message with its location, but with the value of the direction field set to -1; the node is then considered to be active as a *nonstarting node*. Receiving power-on messages may therefore cause nodes to reset their backoff timers or to become inactive. The negative value -1 is used to distinguish power-on messages of volunteering nodes from those of nonvolunteering nodes. The wireless sensor network enters the steady state phase as soon as all nodes changed their state from *undecided* to either being active or inactive. In this last phase the network also performs its main sensing task, until the end of the round. As soon as the round is over, all nodes are switched back to the *undecided* phase, and the OGDC algorithm starts over again.

In our PRTRT specification, each node is represented by an object

```
< l : WSNode | status : s, remainingPower : p, backoffTimer : ti, roundTimer : ti',
               hasVolunteered : v, volunteerProb : r, bitmap : bm, neighbors : nbl >
```

where: l is the node's identifier, representing the coordinates of a point in the Cartesian plane, with the syntax $x\,.\,y$; s can have either one of the values `on`, `off` or `undecided` and represents the node's current status; p denotes the remaining power of the node; ti is a timer counting down to when the node should perform an action, e.g., broadcasting a power-on message; ti' is a timer representing the amount of time left until the end of the current round of the algorithm; v is a Boolean value denoting whether l has volunteered to be a starting node in the current round, or not; r represents the probability that l will volunteer to be a starting node; bm is the node's *bitmap* containing the sections of the node's coverage area that are covered by its neighbors; nbl contains the list of the node's neighbors, i.e., the nodes from which l has received a power-on message in the current round.

The communication model used in the OGDC algorithm is broadcasting with limited transmission range and with transmission delay. It is also assumed that nodes are not aware of their surrounding topology, and therefore a broadcasting node has no information about which of the nodes are going to receive its message. This makes OGDC a fully localized, distributed density control algorithm. A broadcast message has the form `broadcast` m `from` l, where l is the broadcasting node's object identifier and m is the content of the message. In order to model the broadcast to all the other nodes, we follow the approach of [30] and break down each such message into several individual messages addressed to each of the nodes that are within the transmission range of l. A "targeted" message has the form `msg` m `from` l `to` l', where l' is one of the nodes in the range of l. In our case, the message content m may only represent a power-on message, with the syntax `powerOnWithDirection` d, where d is the direction along which the next working node should be located, as explained in [36]. To model transmission delays

of power-on messages, the `dly` operator is applied to power-on messages m using the syntax `dly(`m`,`Δ`)`, denoting that m is delayed by Δ time units before it may become part of the current system configuration. Together with the variable declarations

```
vars R D U : Rat . var B : Bool .  var P : Nat .
vars L L' : Oid .  vars T T' : TimeInf .
var NBS : NeighborList .  vars NB newNeighbor : Neighbor .
vars BM newBitmap : Bitmap .
var CF : Configuration
```

the PRTRT specification of the OGDC algorithm is given in what follows. Each paragraph describes the dynamic behavior of the wireless sensor network at different stages of the algorithm.

Volunteering. The first rule models an "undecided" node which volunteers to be a starting node with probability `R`. If the node volunteers and either its remaining power `P` is greater than a certain value `powerThreshold`, or the probability value `R` is equal to 1, then it sets a backoff timer to a random value `T` which is uniformly distributed on the interval $[0, \texttt{volunteerTimeBound}]$. Otherwise, it sets the timer to a fixed value `nonVolunteerTimer` and doubles its probability of volunteering next time when its backoff timer expires. The Boolean field `hasVolunteered` is also updated correspondingly:

```
prl [volunteer] :
    < L : WSNode | remainingPower : P, volunteerProb : R, hasVolunteered : undecided >
   =>
    if B and (P > powerThreshold or R == 1)
    then < L : WSNode | hasVolunteered : true, backoffTimer : T >
    else < L : WSNode | hasVolunteered : false, backoffTimer : nonVolunteerTimer,
                        volunteerProb : min(2*R, 1) >
    fi
    with probability B := ( true  false ) and T := Uniform(0, volunteerTimeBound) .
                         (  R   1 - R  )
```

If the backoff timer of a node expires, the node did not volunteer, and it also did not receive any power-on messages from other nodes, then the volunteering process is repeated with doubled probability. Note that the probability value `R` has already been doubled in the `[volunteer]` rule, but this rule doubles it once more for its possible further applications, in case the backoff timer is restarted:

```
prl [repeatVolunteering] :
    < L : WSNode | backoffTimer : 0,  hasVolunteered : false, neighbors : none,
                   remainingPower : P, volunteerProb : R >
   =>
    if B and (P > powerThreshold or R == 1)
    then < L : WSNode | backoffTimer : T, hasVolunteered : true >
    else < L : WSNode | backoffTimer : nonVolunteerTimer, volunteerProb : min(2*R, 1) >
    fi
    with probability B := ( true  false ) and T := Uniform(0, volunteerTimeBound) .
                         (  R   1 - R  )
```

Expiration of backoff timers. If the node has volunteered and its backoff timer expires, the node is switched on, its timer is switched off and it broadcasts a power-on message with a random direction uniformly distributed on $[0, 2\pi]$, which consumes some transmission power `transPower`. As soon as this message is broadcast, the node is considered active as a "starting node":

```
prl [startingNodePowerOn] :
    < L : WSNode | backoffTimer : 0, hasVolunteered : true, remainingPower : P >
   =>
    < L : WSNode | status : on, backoffTimer : INF, remainingPower : P monus transPower >
    broadcast (powerOnWithDirection D) from L
    with probability D := Uniform(0, 2 * PI) .
```

If the backoff timer expires and the node did not volunteer, but received a power-on message from at least one other node, also called a "neighbor", then the node switches on, it switches off its timer and broadcasts a power-on message with direction value -1, consuming some transmission power. As soon as this message is broadcast, the node is considered active as a "non-starting node", i.e., one that has been started involuntarily, by other nodes:

```
rl [nonStartingNodePowerOn] :
   < L : WSNode | backoffTimer : 0, hasVolunteered : false, neighbors : NB NBS, remainingPower : P >
  =>
   < L : WSNode | status : on, backoffTimer : INF, remainingPower : P monus transPower >
   broadcast (powerOnWithDirection -1) from L .
```

Receiving power-on messages. The following rules define the dynamics of the network when a node receives a power-on message from another node. If the "undecided" node L receives a power-on message from some other node L' which is within twice the sensing range of L, and if, together with this new node L', the list of neighbors of L fully cover its sensing area, then the node L switches off, stops its backoff timer, adds the new node L' to its list of neighbors and updates its coverage area bitmap. The node L' is added to the list of neighbors of L as a starting or non-starting node, depending on the sign of the value of its direction field:

```
crl [recPowerOnMsgAndSwitchOff] :
    (msg (powerOnWithDirection D) from L' to L)
    < L : WSNode | status : undecided, neighbors : NBS, bitmap : BM >
   =>
    < L : WSNode | status : off, backoffTimer : INF,
                   neighbors : NBS (L' starting (D >= 0)), bitmap : newBitmap >
    if newBitmap := updateBitmap(L, BM, L')
    /\ L withinTwiceTheSensingRangeOf L'
    /\ coverageAreaCovered(newBitmap) .
```

The following three rules apply when the node L', that is sending a power-on message, does not make it so that the coverage area of the receiving node L becomes completely covered. In other words, there exist uncovered crossings formed by intersecting the sensing circles of the neighbors of L. Each of these rules also adds L' to the list of neighbors of L and updates the coverage bitmap of L, by taking into account the sections of its coverage area that are now covered by the new node L'. All rules assume that the "undecided" node L receives a power-on message from another node L' which is within twice the sensing range of L, i.e., $d(\mathtt{L}, \mathtt{L}') \leq 2r_s$. It is also assumed that, by adding the new node L' to the list of neighbors of L, the sensing area of L does not become fully covered by its neighbors. We then have different rules, corresponding to three different cases. In any of these cases, as soon as the backoff timer of L expires, its `status` attribute is changed to `on` and, by application of the `nonStartingNodePowerOn` rule, L will also broadcast a message with the value of the direction field set to -1, to mark that it is a non-starting node.

In the first case, some uncovered crossings are created which fall in the coverage disk of L. Let O be the closest uncovered crossing that lies within the coverage disk of L. If the crossing O is created by L', then L resets its backoff timer to the random value T_a, defined below, which roughly estimates the deviation of L from its optimal position. Otherwise, L leaves the value of its backoff timer unchanged. Notice the use of the `setTa` function, given by means of an equation at the end of this paragraph, in the definition of the variable T' representing the new value of the backoff timer:

```
cprl [recPowerOnMsgWithUncoveredCrossings] :
     (msg (powerOnWithDirection D) from L' to L)
     < L : WSNode | status : undecided, backoffTimer : T, neighbors : NBS, bitmap : BM >
    =>
     < L : WSNode | backoffTimer : T', neighbors : NBS newNeighbor, bitmap : newBitmap >
     if L withinTwiceTheSensingRangeOf L'
     /\ newNeighbor := (L' starting (D >= 0))
     /\ newBitmap := updateBitmap(L, BM, L')
     /\ not coverageAreaCovered(newBitmap)
     /\ existsUncoveredCrossings(L, NBS newNeighbor)
     /\ T' := (if L' createsClosestCrossingTo L withNeighbors (NBS newNeighbor)
               then ceiling(setTa(L, NBS newNeighbor) + U)
               else T
               fi)
     with probability U := Uniform(0, transmissionDelay) .
```

In the second case, no uncovered crossings lie within the coverage disk of L, but at least one of the neighbors of L is a starting node. Under this assumption, if L' is the closest starting neighbor of L, then L resets its backoff timer to the random value T_b defined below; otherwise, it keeps its current value.

Notice the use of the `setTb` function, given by an equation at the end of this paragraph, in the definition of the `T'` variable:

```
cprl [recPowerOnMsgWithStartingNeighbors] :
    (msg (powerOnWithDirection D) from L' to L)
    < L : WSNode | status : undecided, backoffTimer : T, neighbors : NBS, bitmap : BM >
   =>
    < L : WSNode | backoffTimer : T', neighbors : NBS newNeighbor, bitmap : newBitmap >
    if L withinTwiceTheSensingRangeOf L'
    /\ newNeighbor := (L' starting (D >= 0))
    /\ newBitmap := updateBitmap(L, BM, L')
    /\ not coverageAreaCovered(newBitmap)
    /\ not existsUncoveredCrossings(L, NBS newNeighbor)
    /\ existsStartingNeighbor(NBS newNeighbor)
    /\ T' := (if L' == findClosestStartingNeighbor(L, NBS newNeighbor)
              then ceiling(setTb(L, L', D) + U)
              else T
              fi)
    with probability U := Uniform(0, transmissionDelay) .
```

In the third case, no uncovered crossings lie within the sensing range of `L` and none of its neighbors are starting nodes. Under this assumption, if `L'` is the closest neighbor of `L`, then `L` resets its backoff timer to the value T_c, denoted `TC` in our specification; otherwise, it keeps its current value. T_c is a constant which must be greater than $(T_a + T_b)/2$, and much smaller than the value of `nonVolunteerTimer`.

```
crl [recPowerOnMsgWithoutStartingNeighbors] :
    (msg (powerOnWithDirection D) from L' to L)
    < L : WSNode | status : undecided, backoffTimer : T, neighbors : NBS, bitmap : BM >
   =>
    < L : WSNode | backoffTimer : T', neighbors : NBS newNeighbor, bitmap : newBitmap >
    if L withinTwiceTheSensingRangeOf L'
    /\ newNeighbor := (L' starting (D >= 0))
    /\ newBitmap := updateBitmap(L, BM, L')
    /\ not coverageAreaCovered(newBitmap)
    /\ not existsUncoveredCrossings(L, NBS newNeighbor)
    /\ not existsStartingNeighbor(NBS newNeighbor)
    /\ T' := (if L' == findClosestNeighbor(L, NBS newNeighbor)
              then TC
              else T
              fi) .
```

As described in [36], the value of the timer T_a is computed as

$$T_a = t_0\{c[(r_s - d)^2 + (d\Delta\alpha)^2] + u\},$$

where: u is a random variable with an uniform distribution on the interval $[0, 1]$; the variable t_0, called `transmissionDelay` in our specification, denotes the amount of time it takes to send a power-on message; r_s is the sensing range of a node, written `sensingRange` in our specification; $c = 10/r_s^2$ is a constant that determines the backoff scale of timers; d is the distance between the receiving node and the closest uncovered crossing formed by its neighbors, denoted `dTC1(L, NBS)` in our specification; and $\Delta\alpha$ is the angle that the receiving node `L` makes with the closest uncovered crossing and with the optimal location of a new sensor node, written `dAlphaTC1(L, NBS)`. The term $c[(r_s - d)^2 + (d\Delta\alpha)^2]$ roughly represents the deviation of the receiving node from the optimal location, and the value T_a is proportional to this deviation, i.e., the closer the receiving node to the optimal location, the smaller the value of the timer T_a. In this way, nodes that are closer to an optimal location have a higher chance of becoming active. Notice that T_a can also be computed as

$$T_a = t_0 c[(r_s - d)^2 + (d\Delta\alpha)^2] + k,$$

where k is uniformly distributed on $[0, t_0]$. A similar argument holds for the timer value T_b in [36], which can be computed as

$$T_b = t_0 c[(\sqrt{3}r_s - d')^2 + (d'\Delta\beta)^2] + k,$$

with d' the distance between the sending and the receiving nodes, written `dTC2(L, L')` in our specification, and $\Delta\beta$ the angle that the receiving node makes with the sender and with the target direction of the sender, written `dAlphaTC2(L, L', D)`. The following two equations define the `setTa` and `setTb` functions, which in this paper are used to compute the initial values of the backoff timers T_a and T_b, *minus the random term* k, based on the sending and receiving nodes, the list of neighbors `NBS` of the receiving node `L` and the target direction `D` of the sending node `L'`:

```
eq setTa(L, NBS)
  =  transmissionDelay * (c * (
     (sensingRange - dTC1(L, NBS)) * (sensingRange - dTC1(L, NBS))
     + (dTC1(L, NBS) * dTC1(L, NBS) * dAlphaTC1(L, NBS) * dAlphaTC1(L, NBS))
     )) .

eq setTb(L, L', D)
  =  transmissionDelay * (c * (
     (sqrt(3) * sensingRange - dTC2(L, L')) * (sqrt(3) * sensingRange - dTC2(L, L'))
     + dTC2(L, L') * dTC2(L, L') * dAlphaTC2(L, L', D) * dAlphaTC2(L, L', D)
     )) .
```

Discarding power-on messages. If the `status` field of a node `L` has changed from `undecided` to another value, then `L` will ignore any power-on message that it receives from other nodes. In our specification, we also discard messages that are received from nodes `L'` that are not within twice the sensing range of `L`, in order to model broadcasting with limited transmission range:

```
crl [discard] :
    (msg (powerOnWithDirection D) from L' to L)
    < L : WSNode | status : S >
   =>
    < L : WSNode >
    if S =/= undecided or not O withinTwiceTheSensingRangeOf L' .
```

Starting a new round. At the end of each round, i.e., when the `roundTimer` attribute of a node attains a null value, all of the node's attributes, apart from its `remainingPower` attribute, are reinitialized so that a new round of the OGDC algorithm may start:

```
rl [restart] :
   < L : WSNode | roundTimer : 0 >
  =>
   < L : WSNode | status : undecided, neighbors : nil, bitmap : initBitmap(L)
                  hasVolunteered : undecided, volunteerProb : 1.0 / noNodes,
                  backoffTimer : INF, roundTimer : roundTime > .
```

The `initBitmap` function initializes the coverage area bitmap of a node, `noNodes` denotes the total number of nodes in the network and `roundTime` is the amount of time it takes for a round of the OGDC algorithm to complete. These last two parameters can be tuned to different values, to model various situations.

Modeling time elapse. The elapse of time is modeled as in the previous RTT protocol example, by a "standard" tick rule for object-oriented systems

```
crl [tick] : {CF}  =>  {delta(CF, T)} in time T  if T <= mte(CF) .
```

where `delta` is a frozen function that specifies the effect of time elapse on the system, specifically by decreasing the backoff and the round timers of each node, decreasing the remaining power of each node by amounts that correspond to their current status, as well as decreasing the remaining time Δ before each delayed message `dly(`m`,`Δ`)` becomes "ripe" in the current system configuration. The frozen function `mte` gives the maximum amount of time that can elapse before a node must perform an instantaneous transition. More precisely, time cannot advance during the volunteering process of a node or past the expiration of either the backoff timer, or the round timer of a node. Additionally, time cannot advance past the moment when a node runs out of power, to ensure that nodes die exactly when they are supposed to. For a more formal description of the equations defining the above `delta` and `mte` functions we refer the reader to [30].

8 Concluding Remarks

We have defined the probabilistic real-time rewrite theory (PRTRT) formalism for modeling probabilistic real-time systems in rewriting logic, and have shown how PRTRTs can be seen as a unifying semantic framework in which a range of models for probabilistic real-time systems can be naturally represented, including systems with underspecified probability distributions. We have also given a PRTRT specification of a simple round trip time protocol that seems to be outside the class of systems that can be modeled using automaton-based formalisms, since the number of messages in a state can grow beyond any bound.

This work has provided the theoretical foundations for an analysis tool for probabilistic real-time systems in rewriting logic. In the future we must define property specification formalisms and implement suitable model checkers for PRTRTs. For this purpose, the statistical model checking approach seems very promising, since instead of performing exact probabilistic model checking—which often becomes unfeasible for large distributed systems—statistical model checking is typically much more efficient, although it only guarantees a property with a desired level of confidence. In particular, statistical model checking is based on evaluating a number of behaviors and is therefore easily parallelizable. Indeed, the PVeStA tool [4] provides a parallel statistical model checker for a subset of (untimed) probabilistic rewrite theories and could be a useful starting point for a future tool for PRTRTs.

References

1. Agha, G., Greenwald, M., Gunter, C.A., Khanna, S., Meseguer, J., Sen, K., Thati, P.: Formal modeling and analysis of DoS using probabilistic rewrite theories. In: Proc. FCS'05 (2005)
2. Ajmone Marsan, M., Chiola, G.: On Petri nets with deterministic and exponentially distributed firing times. In: Advances in Petri Nets 1987, LNCS, vol. 266. Springer (1987)
3. Ajmone Marsan, M., Conte, G., Balbo, G.: A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. ACM Trans. Comput. Syst. 2 (1984)
4. AlTurki, M., Meseguer, J.: PVeStA: A parallel statistical model checking and quantitative analysis tool. In: Corradini, A., Klin, B., Cîrstea, C. (eds.) CALCO. LNCS, vol. 6859, pp. 386–392. Springer (2011)
5. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. 126 (1994)
6. Bellman, R.: A Markovian decision process. Indiana University Mathematics Journal 6, 679–684 (1957)
7. Bentea, L., Ölveczky, P.C.: Probabilistic real-time rewrite theories and their expressive power. In: Fahrenberg, U., Tripakis, S. (eds.) FORMATS. LNCS, vol. 6919, pp. 60–79. Springer (2011)
8. Bouhoula, A., Jouannaud, J.P., Meseguer, J.: Specification and proof in membership equational logic. Theor. Comput. Sci. 236 (2000)
9. Bruni, R., Meseguer, J.: Semantic foundations for generalized rewrite theories. Theoretical Computer Science 360(1-3) (2006)
10. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.L.: All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic, LNCS, vol. 4350. Springer (2007)
11. D'Argenio, P.R., Katoen, J.P.: A theory of stochastic systems part I: Stochastic automata. Information and Computation 203(1), 1–38 (2005)
12. D'Argenio, P.R., Katoen, J.P., Brinksma, E.: An algebraic approach to the specification of stochastic systems. In: PROCOMET '98. Chapman & Hall, Ltd. (1998)
13. De Alfaro, L.: Formal verification of probabilistic systems. Ph.D. thesis, Stanford University, USA (1998)
14. Gilmore, S., Hillston, J.: The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. In: Computer Performance Evaluation (1994)
15. Johnson, N.L., Kotz, S., Balakrishnan, N.: Continuous Univariate Distributions, Vol. 1 (Wiley Series in Probability and Statistics). Wiley-Interscience, 2 edn. (1994)
16. Katelman, M., Meseguer, J., Hou, J.: Redesign of the LMST wireless sensor protocol through formal modeling and statistical model checking. In: FMOODS'08, LNCS, vol. 5051. Springer (2008)
17. Kendall, D.G.: Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain. The Annals of Mathematical Statistics 24(3), pp. 338–354 (1953)
18. Kumar, N., Sen, K., Meseguer, J., Agha, G.: Probabilistic rewrite theories: Unifying models, logics and tools. Technical report UIUCDCS-R-2003-2347, Department of Computer Science, University of Illinois at Urbana-Champaign (2003)
19. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: Probabilistic model checking for performance and reliability analysis. ACM SIGMETRICS Performance Evaluation Review 36(4) (2009)
20. Kwiatkowska, M., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. Theor. Comput. Sci. 282 (2002)
21. Lien, E., Ölveczky, P.C.: Formal modeling and analysis of an IETF multicast protocol. In: SEFM'09. IEEE Computer Society (2009)
22. Manna, Z., Pnueli, A.: Models for reactivity. Acta Informatica 30, 609–678 (1993)

23. Meseguer, J.: Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science 96(1) (1992)
24. Meseguer, J.: Membership algebra as a logical framework for equational specification. In: Recent Trends in Algebraic Development Techniques, LNCS, vol. 1376. Springer (1998)
25. Meseguer, J., Montanari, U.: Petri nets are monoids. Information and Computation 88(2) (1990)
26. Ölveczky, P.C., Meseguer, J.: Specification of real-time and hybrid systems in rewriting logic. Theoretical Computer Science 285(2) (2002)
27. Ölveczky, P.C., Meseguer, J.: Semantics and pragmatics of Real-Time Maude. Higher-Order and Symbolic Computation 20 (2007)
28. Ölveczky, P.C., Meseguer, J.: The Real-Time Maude Tool. In: TACAS'08. pp. 332–336 (2008)
29. Ölveczky, P.C., Meseguer, J., Talcott, C.: Specification and analysis of the AER/NCA active network protocol suite in Real-Time Maude. Formal Methods in System Design 29 (2006)
30. Ölveczky, P.C., Thorvaldsen, S.: Formal modeling and analysis of the OGDC wireless sensor network algorithm in Real-Time Maude. In: Bonsangue, M., Johnsen, E. (eds.) Formal Methods for Open Object-Based Distributed Systems, Lecture Notes in Computer Science, vol. 4468, pp. 122–140. Springer Berlin / Heidelberg (2007)
31. Sen, K., Viswanathan, M., Agha, G.A.: VESTA: A statistical model-checker and analyzer for probabilistic systems. In: QEST'05. IEEE Computer Society (2005)
32. Sproston, J.: Model Checking for Probabilistic Timed and Hybrid Systems. Ph.D. thesis, School of Computer Science, University of Birmingham (2001)
33. Stehr, M.O., Meseguer, J., Ölveczky, P.C.: Rewriting logic as a unifying framework for petri nets. In: Unifying Petri Nets, LNCS, vol. 2128. Springer (2001)
34. Thorvaldsen, S., Ölveczky, P.C.: Formal modeling and analysis of the OGDC wireless sensor network algorithm in Real-Time Maude (Oct 2005), manuscript
35. Yi, W.: Algebraic reasoning for real-time probabilistic processes with uncertain information. In: FTRTFT'94, LNCS, vol. 863. Springer (1994)
36. Zhang, H., Hou, J.: Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. Ad Hoc & Sensor Wireless Networks 1(1-2) (2005)

A Full Specification of the RTT Example

We provide the full PRTRT specification of our probabilistic version of the RTT protocol, using Maude syntax. The definition of the function `F(x)`, representing the probability distribution of the message transmission delays, depends on the implementation of functions that compute numerical approximations of the probability density function (PDF) and the cumulative distribution function (CDF) of the standard normal distribution. There are several numerical algorithms in the literature computing these functions, and we therefore omit their implementation.

```
--- Variable declarations
vars O O' : Oid .  var M : Msg .  var TI : TimeInf .  vars T T' : Time .
var CF : Configuration .  var B : Bool .  var N : Nat .  var X : Float .

--- Definition of the delay message operator dly(M, D)
--- which has right identity 0, i.e., dly(M, 0) = M for all messages M.
sort DlyMsg .
subsort Msg < DlyMsg .
op dly : Msg Time -> DlyMsg [ctor right id: 0] .

--- Definition of the parameters (mean, standard deviation, lower bound)
--- of the truncated normal probability distribution followed
--- by the message transmission delay. These parameters can be tweaked
--- to match the system that we are modeling.
ops distMean distDev minDelay : -> Float [ctor] .

--- Messages (either requests or response) take 10 time units to be sent, on average.
eq distMean = 10.0 .

--- The standard deviation of the transmission delay from the mean value of 10 is 0.5.
eq distDev = 0.5 .

--- The lower bound for the transmission delay of a message in the network
eq minDelay = 10.0 .

--- Definition of the maximum allowed round trip time,
--- before node O should send another RTT request to O'.
eq maxRTT = 20.0 .

--- Definition of the dist function, computing the Euclidean distance
--- between two nodes. We assume that each object contains the attributes
--- XPos and YPos, representing the coordinates of the node.
vars O1 O2 : Oid .  vars X1 X2 Y1 Y2 : Float .
op dist : Object Object -> Float [ctor] .
eq dist(< O1 : Node | XPos : X1, YPos : Y1 >,
        < O2 : Node | XPos : X2, YPos : Y2 >)
  = sqrt((X1 - X2) * (X1 - X2) + (Y1 - Y2) * (Y1 - Y2)) .

--- We next define the probability distribution function F(X)
--- that mimics the probability density function (PDF)
--- of a truncated normal distribution with mean distMean,
--- standard deviation distDev, and lower bound minDelay.
--- The value of F(X) is therefore zero for X < minDelay,
--- and its plot looks like a truncated bell shape curve for X >= minDelay .
--- The functions PDFNorm and CDFNorm compute the PDF
--- and the CDF of the standard normal distribution, respectively.
op F : Float -> Float [ctor] .
eq F(X) = PDFNorm((X - distMean)/distDev)
        / (distDev * (1 - CDFNorm((minDelay - distMean) / distDev))) .


--- This rule starts the RTT protocol by sending an RTT request message
--- from node O to node O'. The request becomes ripe in the configuration
--- after a random, normally distributed amount of time has passed.
--- The retransmission timer of O is also set to expire in time maxRTT .
prl [start] :
    findRtt(O)  < O : Node | clock : T, nbr : O' >  =>
                < O : Node | timer : maxRTT >  dly(rttReq(O', O, T), D)
    with probability D := F(dist(O, O')) .
```

```
--- This rule specifies how node O' responds to a request from O.
--- Namely, O' sends a delayed response message to O with probability 3/4,
--- or drops the request from O with probability 1/4.
--- The delay of the response message is distributed
--- according to the same probability distribution
--- as in the case of request messages from O.
prl [rttResp] :
    rttReq(O, O', T)  < O : Node | >  =>
    if B then < O : Node | >  dly(rttResp(O', O, T), D)
         else < O : Node | >
    fi
    with probability B := ( true  false ) and D := F(dist(O, O')) .
                          ( 3/4   1/4   )

--- The following conditional rule models the situation in which
--- O receives a response from O' and manages to compute the RTT value.
--- The rule is only applied if the total waiting time of O
--- is less than the constant maxRTT.
crl [treatResp] :
    rttResp(O, O', T)  < O : Node | clock : T' >  =>
                       < O : Node | rtt : T' - T, timer : INF >
    if T' - T < maxRTT .

--- If the response from O' comes too late, i.e., if the total round trip time
--- of O would be greater than maxRTT, then O ignores the response message.
crl [ignoreOld] :
    rttResp(O, O', T)  < O : Node | clock : T' >  =>  < O : Node | >
    if T' - T >= maxRTT .

--- If the retransmission timer of O expires, then O sends another
--- RTT request to O' with probability 1/(N + 1), where N is the number of
--- (unsuccessful) attempts of O to compute its RTT value.
--- Therefore, the probability of starting a new round of the RTT protocol
--- decreases with the number of unsuccessful attempts of O.
--- If O does not start another round of the RTT protocol,
--- then it resets its 'tries' attribute to 0 and becomes idle.
prl [tryAgain] :
    < O : Node | timer : 0, clock : T, nbr : O', tries : N >  =>
    if B then
      < O : Node | timer : maxRTT, tries : N + 1 >  dly(rttReq(O, O', T), D)
    else < O : Node | timer : maxRTT >
    fi
    with probability B := ( true    false   ) and D := F(dist(O, O')) .
                          ( 1/(N+1) N/(N+1) )

--- The following rule is the main tick rule for our system
--- and specifies how the system evolves with the passing of time.
crl [tick] :
    { CF }  =>  { delta(CF, T) } in time T  if T <= mte(CF) .

--- We give below the definitions for the delta and mte functions.
--- Their application on delayed messages is also defined.
--- The maximum time elapse (mte) function is defined such that time cannot advance
--- past the moment when a message becomes ripe in the current configuration.
--- This forces ripe messages to be treated immediately.

--- Recursive definition of the delta function,
--- describing the effect of time elapse on the system configuration
op delta : Configuration Time -> Configuration [frozen (1)] .
eq delta(< O : Node | timer : TI, clock : T > CF, T')
  = < O : Node | timer : TI - T', clock : T + T' > delta(CF, T') .
eq delta(dly(M, T) CF, T') = dly(M, T - T') delta(CF, T') .
eq delta(CF) = CF [owise] .

--- Recursive definition of the mte function,
--- describing the maximum amount of time that can elapse
--- before an action (instantaneous transition) takes place.
op mte : Configuration -> TimeInf [frozen (1)] .
eq mte(< O : Node | timer : TI > CF) = min(TI, mte(CF)) .
eq mte(dly(M, T) CF) = min(T, mte(CF)) .
eq mte(CF) = INF [owise] .
```

B PRTRT Representation of DSPNs

We give a more detailed specification, using Maude syntax, of the PRTRT representation of a DSPN.

```
sort Place .
ops p1 p2 p3 p4 ... : -> Place [ctor] . --- one constant for each place

sort Transition .
ops t1 t2 t3 ... : -> Transition [ctor] . --- one constant for each transition

op reset : -> TimeInf [ctor] . --- new 'timer' value

sort TransitionTimer .
op <_;_> : Transition TimeInf -> TransitionTimer [ctor] .

--- An extended marking is a multiset of places and transition timers:
sort ExtendedMarking .
subsort TransitionTimer Place < ExtendedMarking .
op none : -> ExtendedMarking [ctor] . --- empty marking

--- associative-commutative multiset union operator
op __ : ExtendedMarking ExtendedMarking -> ExtendedMarking [ctor assoc comm id: none] .

--- For each transition t we have a firing rule of the following form.
--- Assume that the preset is q1 ... qm and the postset is q1' ... qn',
--- where each qi and qk' is some pj:
vars REST : ExtendedMarking .  var T : Transition .
var TI : TimeInf .  var X : Time .

rl [fire-t] :
   { < t ; 0 >  q1 ... qm REST }  =>
   { recomputeTimers(< t ; INF >  q1' ... qn' REST) } .


op recomputeTimers : ExtendedMarking -> ExtendedMarking [frozen (1)] .

--- For each deterministic transition t, with the above pre- and postsets,
--- there is one equation as follows:
eq recomputeTimers(< t ; TI >  q1 ... qm REST)  --- t is enabled
  = if TI == INF         --- t was previously disabled
    then < t ; tau(t) >  --- initialize with value of firing delay
    else < t ; TI >      --- t was already enabled, do not change the timer value
    fi
    recomputeTimers(q1 ... qm REST) .  --- recursively compute the other timers

--- For each stochastic transition t, with the above pre- and postsets,
--- there is one equation as follows, which is very similar
--- to the above equation, but resets the timer to reset:
eq recomputeTimers(< t ; TI >  q1 ... qm REST)  --- t is enabled
  = if TI == INF        --- t was previously disabled
    then < t ; reset >  --- initialize with value reset
    else < t ; TI >     --- t was already enabled, do not change timer value
    fi
    recomputeTimers(q1 ... qm REST) .  --- recursively compute the other timers

--- An 'owise' equation matches when the transition T is not enabled.
--- Then we set the timer to INF, no matter its earlier value:
eq recomputeTimers(< T ; TI >  REST)  =  < T ; INF >  recomputeTimers(REST) .

--- When there are no transition-timer pairs left to apply recomputeTimers to, we are finished:
ceq recomputeTimers(REST)  =  REST  if noTimers(REST) .

--- The noTimers operator could also have been specified with sorts, etc.
op noTimers : ExtendedMarking -> Bool .
eq noTimers(< T ; TI >  REST)  = false .
eq notimers(REST)  =  true [owise] .
```

```
--- Next, we instantiate the timer variable X probabilistically.
--- (Details about the probability measure PI are omitted.)
rl [instantiate-probabilistic-delay] :
   < T ; reset >  =>  < T ; X >  with probability X := PI( ... R(T) ... ) .

--- Finally, the tick rule:
var SYSTEM : ExtendedMarking .
crl [tick] :
    { SYSTEM }  =>  { decreaseTimers(SYSTEM, X) }  in time X if X <= mte(SYSTEM) .

--- Decrease the timers of all transition-timer pairs in the given extended marking
--- by the amount specified in the second argument:
op decreaseTimers : ExtendedMarking Time -> ExtendedMarking [frozen (1)] .
eq decreaseTimers(< T ; TI >  REST, X)
  = < T ; TI monus X >  decreaseTimers(REST, X) .
eq decreaseTimers(REST) = REST [owise] .  --- no more timers

--- The function mte gives the smallest timer value in the system:
op mte : ExtendedMarking -> TimeInf [frozen (1)] .
eq mte(< T ; TI >  REST)
  = min(if TI == reset then 0 else TI fi, mte(REST)) .
eq mte(REST) = INF [owise] .
```